```
SQL> /

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GO$SQL_BIND_CAPTURE
select INST_ID,              KQLFBC_PADD,              KQLFBC_HASH,
KQLFBC_SQLID,              KQLFBC_CADD,              KQLFBC_CHNO,
substr(KQLFBC_NAME, 1, 30),              KQLFBC_POS,
to_number(decode(KQLFBC_DUPPOS, 65535, NULL, KQLFBC_DUPPOS)),
KQLFBC_OACDTY,              substr(KQLFBC_DTYSTR, 1, 15),
decode(KQLFBC_OACCSI, 0, to_number(null), KQLFBC_OACCSI),
decode(KQLFBC_OACPRE, 0, to_number(null), KQLFBC_OACPRE),
decode(KQLFBC_OACSCL, 0, to_number(null), KQLFBC_OACSCL),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
KQLFBC_OACMXL,              decode(KQLFBC_WCAP, 0, 'NO', 'YES'),
decode(KQLFBC_WCAP, 0, to_date(NULL),                  decode(KQLFBC_WCAP,
2, to_date(NULL), KQLFBC_LCAP)),              KQLFBC_STRVAL,
decode(KQLFBC_WCAP, 0, NULL,
sys.sys$rawtoany(KQLFBC_BINVAL, KQLFBC_OACDTY,
KQLFBC_OACCSF, KQLFBC_OACCSI))         from x$kqlfbc

GV$ACCESS
select distinct s.inst_id,s.ksusenum,o.kglnaown,o.kglnaobj, decode(o.kglobtyp,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
0, 'CURSOR',    1, 'INDEX',    2, 'TABLE',    3, 'CLUSTER',    4, 'VIEW',    5,
'SYNONYM',    6, 'SEQUENCE',    7, 'PROCEDURE',    8, 'FUNCTION',    9,
'PACKAGE',    10,'NON-EXISTENT',    11,'PACKAGE BODY',    12,'TRIGGER',
13,'TYPE',    14,'TYPE BODY',    15,'OBJECT',    16,'USER',    17,'DBLINK',
18,'PIPE',    19,'TABLE PARTITION',    20,'INDEX PARTITION',    21,'LOB',
22,'LIBRARY',    23,'DIRECTORY',    24,'QUEUE',    25,'INDEX-ORGANIZED TABLE',
26,'REPLICATION OBJECT GROUP',    27,'REPLICATION PROPAGATOR',    28,'JAVA
SOURCE',    29,'JAVA CLASS',    30,'JAVA RESOURCE',    31,'JAVA JAR',
'INVALID TYPE') from x$ksuse s,x$kglob o,x$kgldp d,x$kgllk l where

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
l.kgllkuse=s.addr and l.kgllkhdl=d.kglhdadr and l.kglnahsh=d.kglnahsh and
o.kglnahsh=d.kglrfhsh and o.kglhdadr=d.kglrfhdl

GV$ACTIVE_INSTANCES
```

```
select inst_id, ksiminum, rpad(ksimstr,60) from x$ksimsi
```

GV$ACTIVE_SERVICES
```
select inst_id, kswsastabsi, kswsastabnm, kswsastabnmh, kswsastabnn,
kswsastabcd, kswsastabcdh,  decode(kswsastabgoal, -1, NULL, 0, 'NONE', 1,
```

VIEW_NAME

VIEW_DEFINITION

```
'SERVICE_TIME', 2, 'THROUGHPUT', NULL) kswsastabgoal,
decode(bitand(kswsastabpflg, 2), 2, 'Y', 'N'),  decode(bitand(kswsastabgflg, 1),
1, 'YES', 'NO'),  decode(bitand(kswsastabpflg, 4), 4, 'YES', 'NO'),
decode(bitand(kswsastabpflg, 8), 8, 'LONG', 'SHORT')  from x$kswsastab  where
kswsastabact = 1
```

GV$ACTIVE_SESSION_HISTORY
```
SELECT  /*+ no_merge ordered use_nl(s,a) */ a.inst_id, s.sample_id,
s.sample_time, a.session_id, a.session_serial#, decode(a.session_type, 1,
```

VIEW_NAME

VIEW_DEFINITION

```
'FOREGROUND', 'BACKGROUND'), a.flags, a.user_id, a.sql_id, a.sql_child_number,
a.sql_opcode, a.force_matching_signature, decode(a.top_level_sql_id, NULL,
a.sql_id, a.top_level_sql_id), decode(a.top_level_sql_id, NULL, a.sql_opcode,
a.top_level_sql_opcode), a.sql_plan_hash_value, decode(a.sql_plan_line_id, 0,
to_number(NULL),       a.sql_plan_line_id), a.sql_plan_operation,
a.sql_plan_options, decode(a.sql_exec_id, 0, to_number(NULL), a.sql_exec_id),
a.sql_exec_start, decode(a.plsql_entry_object_id,0,to_number(NULL),
a.plsql_entry_object_id), decode(a.plsql_entry_object_id,0,to_number(NULL),
a.plsql_entry_subprogram_id),
```

VIEW_NAME

VIEW_DEFINITION

```
decode(a.plsql_object_id,0,to_number(NULL),a.plsql_object_id),
decode(a.plsql_object_id,0,to_number(NULL),a.plsql_subprogram_id),
decode(a.qc_session_id, 0, to_number(NULL), a.qc_instance_id),
decode(a.qc_session_id, 0, to_number(NULL), a.qc_session_id),
decode(a.qc_session_id, 0, to_number(NULL), a.qc_session_serial#),
decode(a.wait_time, 0, a.event,    NULL), decode(a.wait_time, 0, a.event_id,
NULL), decode(a.wait_time, 0, a.event#,   NULL), a.seq#, a.p1text, a.p1,
a.p2text, a.p2, a.p3text, a.p3, decode(a.wait_time, 0, a.wait_class,    NULL),
decode(a.wait_time, 0, a.wait_class_id, NULL), a.wait_time, decode(a.wait_time,
```

VIEW_NAME

VIEW_DEFINITION

```
0, 'WAITING', 'ON CPU'), a.time_waited, (case when a.blocking_session =
4294967295        then 'UNKNOWN'       when a.blocking_session = 4294967294
then 'GLOBAL'       when a.blocking_session = 4294967293        then 'UNKNOWN'
when a.blocking_session = 4294967292        then 'NO HOLDER'       when
a.blocking_session = 4294967291        then 'NOT IN WAIT'       else 'VALID'
end), (case when a.blocking_session between 4294967291 and 4294967295
then to_number(NULL)       else a.blocking_session  end), (case when
a.blocking_session between 4294967291 and 4294967295        then
to_number(NULL)       else a.blocking_session_serial#  end), a.current_obj#,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
a.current_file#, a.current_block#, a.current_row#, decode(a.consumer_group_id,
0, to_number(NULL), a.consumer_group_id), a.xid, decode(a.remote_instance#, 0,
to_number(NULL), a.remote_instance#), a.in_connection_mgmt, a.in_parse,
a.in_hard_parse, a.in_sql_execution, a.in_plsql_execution, a.in_plsql_rpc,
a.in_plsql_compilation, a.in_java_execution, a.in_bind, a.in_cursor_close,
a.service_hash, a.program, a.module, a.action, a.client_id FROM  x$kewash s,
x$ash a WHERE s.sample_addr = a.sample_addr and       s.sample_id   =
a.sample_id   and       s.sample_time = a.sample_time


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$ACTIVE_SESS_POOL_MTH
select inst_id, policy_name_kgskasp from x$kgskasp

GV$ADVISOR_PROGRESS
select inst_id, ksulosno, ksulosrn, ksulounm, ksulopna, ksuloif5d,
ksuloctx,  ksulotde, ksulosfr, ksulotot, ksulouni,
ksuloif3, ksuloif4, ksuloif0,  ksuloif1, ksuloif2,
to_date(ksulostm,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),
to_date(ksulolut,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
ksuloetm,
decode(ksuloif6d, null, to_number(null), ksuloif6), ksuloif6d,
ksuloif7d                                              from
x$ksulop                                                where
ksulopna in ('Advisor')

GV$ALERT_TYPES
SELECT t.inst_id, rid_keltsd, typnam_keltosd,          decode(typ_keltsd, 1,
'Stateful', 'Stateless'), nam_keltgsd,          decode(scp_keltsd, 1,
```

VIEW_NAME

VIEW_DEFINITION

'Database', 'Instance'), mtn_keltsd, mtc_keltsd FROM x$keltsd t, x$keltosd, x$keltgsd WHERE grp_keltsd = id_keltgsd AND otyp_keltsd = typid_keltosd

GV$AQ1
select INST_ID, KWQSIQID , KWQSINWT, KWQSINRD, KWQSINEX, KWQSINCO, KWQSITWT, DECODE(KWQSINCO, 0, 0, KWQSITWT/KWQSINCO) from X$KWQSI

VIEW_NAME

VIEW_DEFINITION

GV$ARCHIVE
select le.inst_id,le.lenum,le.lethr,le.leseq, decode(bitand(le.leflg,8),0,'NO','YES'), decode(bitand(le.leflg,8),0,'NO','YES'), to_number(le.lelos) from x$kccle le,x$kccdi di where bitand(di.diflg,1)!=0 and le.ledup!=0 and bitand(le.leflg,1)=0 and (to_number(le.lelos)<=to_number(di.difas) or bitand(le.leflg,8)=0)

GV$ARCHIVED_LOG

VIEW_NAME

VIEW_DEFINITION
---
select inst_id,alrid,alstm,alnam,aldst,althp,alseq,to_number(alrls),to_date(alrlc,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(alxlc),to_number(allos),to_date(allot,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(alnxs),to_date(alnxt,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),albct,albsz,decode(bitand(alflg, 16+32+64+128+256), 16, 'ARCH', 32, 'FGRD', 64, 'RMAN', 128,'SRMN', 256,'LGWR', 'UNKNOWN'),decode(bitand(alflg, 4), 4, 'RFS',

VIEW_NAME

VIEW_DEFINITION
---
decode(bitand(alflg, 16+32+64+128+256), 16, 'ARCH', 32, 'FGRD', 64, 'RMAN', 128,'SRMN', 256,'LGWR', 'UNKNOWN')),decode(bitand(alflg, 8),0,'NO','YES'),decode(bitand(alflg, 2),0,'NO','YES'),decode(bitand(alflg, 1024),0,'NO','YES'),decode(bitand(alflg, 1),0,'NO','YES'),decode(bitand(alflg, 1+2048+4096), 0, 'A', 1, 'D', 2048,'X', 4096,'U', '?'),to_date(altsm,'MM/DD/RR

```
HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(bitand(alflg,8192),0,'NO','YES'),de
code(bitand(alflg,16384),0,'NO','YES'),
decode(bitand(alflg,32768),0,'NO','YES'), to_number(bitand(alfl2,15)),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
altoa,alacd,decode(bitand(alfl2,64),0,'NO','YES'),
decode(bitand(alfl2,128),0,'NO','YES'), decode(bitand(alflg,512),0,'NO','YES'),
decode(bitand(alfl2,256+512+1024),      256, 'TERMINAL',      512,
'ACTIVATION',     1024, 'RESETLOGS',
decode(bitand(alflg,32768),0,'','SWITCHOVER')), decode(bitand(alfl2,4096), 0,
'NO', 'YES') from x$kccal

GV$ARCHIVE_DEST
select inst_id, to_number(ADDID), ADDXX, decode(ADSTS,1,'VALID',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
2,'INACTIVE',             3,'DEFERRED',             4,'ERROR',
5,'DISABLED',             6,'BAD PARAM',            7,'ALTERNATE',
8,'FULL',             'UNKNOWN'), decode(ADMND,0,'OPTIONAL','MANDATORY'),
decode(ADSES,0,'SYSTEM','SESSION'), decode(ADRMT,0,'PRIMARY',
1,'STANDBY',              2,'LOCAL',            3,'REMOTE',
'UNKNOWN'), decode(ADPRC,0,'ARCH',            1,'LGWR',
2,'FOREGROUND',           3,'RFS',            4,'ONDEMAND',
'UNKNOWN'), decode(ADSCH,0,'INACTIVE',            1,'PENDING',
2,'ACTIVE',             3,'LATENT',             'UNKNOWN'), decode(ADOMF, 0,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
ADDNM, 'USE_DB_RECOVERY_FILE_DEST'), to_number(ADLSQ), to_number(ADROP),
to_number(ADDLY), to_number(ADMCS), to_number(ADNTT), decode(ADWHO,0,'ARCH',
1,'LGWR',             2,'FOREGROUND',             3,'RFS',
4,'ONDEMAND',             'UNKNOWN'), decode(ADREG,0,'NO','YES'),
to_date(ADFDT,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'), to_number(ADFSQ),
to_number(ADFBK), to_number(ADFCT), to_number(ADMXF), ADFER, ADALT, ADDPD,
ADRFT, to_number(ADQSZ), to_number(ADQSD), ADMID, decode(ADLAB,0,
decode(ADPAR,0,'SYNCHRONOUS',                     'PARALLELSYNC'),
'ASYNCHRONOUS'), to_number(ADLAB), decode(ADAFF,0,'NO','YES'),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(ADDTG,0,'PUBLIC','PRIVATE'), decode(ADNOW,1,'YES',             2,'WRONG
VALID_TYPE',             3,'WRONG VALID_ROLE',             4,'INACTIVE',
```

```
'UNKNOWN'), decode(ADVLD,11,'ONLINE_LOGFILE',            12,'ONLINE_LOGFILE',
13,'ONLINE_LOGFILE',            21,'STANDBY_LOGFILE',
22,'STANDBY_LOGFILE',            23,'STANDBY_LOGFILE',
31,'ALL_LOGFILES',            32,'ALL_LOGFILES',
33,'ALL_LOGFILES',            'UNKNOWN'), decode(ADVLD,11,'PRIMARY_ROLE',
12,'STANDBY_ROLE',            13,'ALL_ROLES',            21,'PRIMARY_ROLE',
22,'STANDBY_ROLE',            23,'ALL_ROLES',            31,'PRIMARY_ROLE',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
32,'STANDBY_ROLE',            33,'ALL_ROLES',            'UNKNOWN'),
ADDBUN, decode(ADVER,0,'NO','YES'), decode(ADCMP,0,'DISABLE','ENABLE') from
x$krstdest

GV$ARCHIVE_DEST_STATUS
select inst_id, to_number(DSDID), DSDXX, decode(DSSTS,1,'VALID',
2,'INACTIVE',            3,'DEFERRED',            4,'ERROR',
5,'DISABLED',            6,'BAD PARAM',            'UNKNOWN'),
decode(DSTYP,1,'LOCAL',            2,'PHYSICAL',            3,'LOGICAL',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
4,'CROSS-INSTANCE',            5,'SNAPSHOT',            6,'DOWNSTREAM',
'UNKNOWN'), decode(DSDMD,1,'STARTED',            2,'MOUNTED',
3,'MOUNTED-STANDBY',            4,'OPEN',            5,'OPEN_READ-ONLY',
'UNKNOWN'), decode(DSRMD,1,'IDLE',            2,'MANUAL',
3,'MANAGED',            4,'MANAGED REAL TIME APPLY',            5,'LOGICAL
REAL TIME APPLY',            'UNKNOWN'), decode(DSPRT,0,'MAXIMUM
PERFORMANCE',            1,'MAXIMUM PROTECTION',            2,'MAXIMUM
AVAILABILITY',            3,'RESYNCHRONIZATION',
'UNKNOWN'),DSDNM, to_number(DSCNT), to_number(DSACT), to_number(DSLTA),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
to_number(DSLSA), to_number(DSLTR), to_number(DSLSR), DSERR,
decode(DSSRL,0,'NO',            'YES'), DSDBUN, decode(DSPRC, 1, 'CHECK
CONFIGURATION',            2, 'CHECK STANDBY REDO LOG',            3,
'CHECK CONNECTIVITY',            4, 'DESTINATION HAS A GAP',            5,
'OK',            'STATUS NOT AVAILABLE'), decode(DSPRT, 1, 'YES',
2, 'YES',            'NO')  from x$kcrrdstat

GV$ARCHIVE_GAP
select USERENV('Instance'), high.thread#, low.lsq, high.hsq  from   (select

VIEW_NAME
------------------------------
```

VIEW_DEFINITION
---
a.thread#, rcvsq, min(a.sequence#)-1 hsq from v$archived_log a, (select thread#, max(sequence#) rcvsq from v$log_history group by thread#) b where a.thread# = b.thread# and a.sequence# > rcvsq group by a.thread#, rcvsq) high, (select thread#, min(sequence#)+1 lsq from v$log_history, v$datafile where checkpoint_change# <= next_change# and checkpoint_change# >= first_change# and enabled = 'READ WRITE' group by thread#) low where low.thread# = high.thread# and lsq < = hsq and hsq > rcvsq

GV$ARCHIVE_PROCESSES

VIEW_NAME

VIEW_DEFINITION
---
select inst_id, to_number(kcrrxpid), decode(kcrrxsts, 1,'SCHEDULED', 2,'STARTING', 3,'ACTIVE', 4,'STOPPING', 5,'TERMINATED', 6,'INITING', 'STOPPED'), to_number(kcrrxseq), decode(kcrrxsta, 1,'BUSY', 'IDLE') from x$kcrrarch

GV$ASM_ALIAS
select inst_id, name_kfals, group_kfals, number_kfals, incarn_kfals, entnum_kfals, entinc_kfals, parent_kfals, refer_kfals, decode(bitand(entflg_kfals, 12), 4, 'Y', 8, 'Y', 'N'),

VIEW_NAME

VIEW_DEFINITION
---
decode(bitand(entflg_kfals, 15), 1, 'N', 2, 'Y', 4, 'Y', 8, 'N') from x$kfals where decode(bitand(entflg_kfals, 12), 4, 'Y', 8, 'Y', 'N') = 'Y' OR decode(bitand(entflg_kfals, 16), 16, 'Y', 'N') = 'Y'

GV$ASM_ATTRIBUTE
select inst_id, name_kfenv, value_kfenv, group_kfenv, entnum_kfenv, entinc_kfenv, decode(readonly_kfenv, 0, 'N', 1, 'Y', 'UNKNOWN') readonly_kfenv, decode(system_kfenv, 0, 'N', 1, 'Y', 'UNKNOWN') system_kfenv from x$kfenv where hidden_kfenv = 0

VIEW_NAME

VIEW_DEFINITION

GV$ASM_CLIENT
select inst_id, gn_kfncl, instname_kfncl, dbname_kfncl, decode(status_kfncl, 1, 'CONNECTED', 2, 'DISCONNECTED', 3, 'BROKEN'), softver_kfncl, compver_kfncl from x$kfncl

GV$ASM_DISK
select d.inst_id, d.grpnum_kfdsk, d.number_kfdsk, d.compound_kfdsk,

d.incarn_kfdsk, decode(d.mntsts_kfdsk, 1, 'MISSING', 2, 'CLOSED', 3,

VIEW_NAME

VIEW_DEFINITION

'OPENED', 4, 'CACHED', 5, 'IGNORED', 6, 'IGNORED', 7, 'CLOSING', 8,
'IGNORED', 9, 'IGNORED', 'INVALID'), decode(d.hdrsts_kfdsk, 1,
'UNKNOWN', 2, 'CANDIDATE', 3, 'MEMBER', 4, 'FORMER', 5, 'CONFLICT', 6,
'INCOMPATIBLE', 7, 'PROVISIONED', 8, 'FOREIGN', 'INVALID'),
decode(d.grpnum_kfdsk, 0, 'ONLINE', decode(d.mode_kfdsk, 0, 'UNKNOWN', 127,
'ONLINE', 21, 'OFFLINE', 1, 'OFFLINE', 'SYNCING')), decode(d.state_kfdsk,
1, 'UNKNOWN', 2, 'NORMAL', 3, 'UNUSED', 4, 'DROPPING', 5,
decode(d.mode_kfdsk, 127, 'DROPPING', 'FORCING'), 6, 'FORCING', 7, 'DROPPED',
8, 'ADDING', 'INVALID'), decode(d.redun_kfdsk, 16, 'UNPROT', 17, 'UNPROT',

VIEW_NAME

VIEW_DEFINITION

18, 'MIRROR', 19, 'MIRROR', 20, 'MIRROR', 21, 'MIRROR', 22, 'MIRROR', 23,
'MIRROR', 32, 'PARITY', 33, 'PARITY', 34, 'PARITY', 35, 'PARITY', 36,
'PARITY', 37, 'PARITY', 38, 'PARITY', 39, 'PARITY', 'UNKNOWN'),
d.libnam_kfdsk, d.osmb_kfdsk, d.totmb_kfdsk, d.totmb_kfdsk - d.usedmb_kfdsk,
d.asmname_kfdsk, d.failname_kfdsk, d.label_kfdsk, d.path_kfdsk, d.udid_kfdsk,
d.product_kfdsk, d.crdate_kfdsk, d.mtdate_kfdsk, d.timer_kfdsk,
k.read_kfkid, k.write_kfkid, k.rerr_kfkid, k.werr_kfkid,
k.rtime_kfkid/1000000, k.wtime_kfkid/1000000, k.bytesr_kfkid, k.bytesw_kfkid,
decode(d.prefrd_kfdsk, 0, ' ', 1, 'Y', 2, 'N', 3, 'U') from x$kfdsk d,

VIEW_NAME

VIEW_DEFINITION
----
x$kfkid k where d.mntsts_kfdsk != 0 and d.kfkid_kfdsk = k.idptr_kfkid(+)

GV$ASM_DISKGROUP
select g.inst_id, g.number_kfgrp, g.name_kfgrp, g.sector_kfgrp,
g.blksize_kfgrp, g.ausize_kfgrp, decode(g.state_kfgrp, 0, 'INVALID', 1,
'UNKNOWN', 2, 'DISMOUNTED', 3, 'CREATING', 4, 'MOUNTING', 5, 'MOUNTED', 6,
'DISMOUNTING', 7, 'CONNECTED', 8, 'BROKEN', 9, 'CONNECTING', 10, 'BREAKING',
11, 'DROPPING', 12, 'DROPPING', 255, 'RESTRICTED'), decode(g.type_kfgrp,
1, 'EXTERN', 2, 'NORMAL', 3, 'HIGH'), g.totmb_kfgrp, g.freemb_kfgrp,

VIEW_NAME

VIEW_DEFINITION
----
g.minspc_kfgrp, g.usable_kfgrp, g.offline_kfgrp, g.compat_kfgrp,
g.dbcompat_kfgrp from x$kfgrp g where state_kfgrp != 0

GV$ASM_DISKGROUP_STAT

```
select  g.inst_id, g.number_kfgrp, g.name_kfgrp,  g.sector_kfgrp,
g.blksize_kfgrp, g.ausize_kfgrp,  decode(g.state_kfgrp,  0, 'INVALID', 1,
'UNKNOWN', 2, 'DISMOUNTED', 3, 'CREATING',  4, 'MOUNTING', 5, 'MOUNTED', 6,
'DISMOUNTING', 7, 'CONNECTED',  8, 'BROKEN', 9, 'CONNECTING', 10, 'BREAKING',
11, 'DROPPING',  12, 'DROPPING', 255, 'RESTRICTED'),  decode(g.type_kfgrp,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
1, 'EXTERN', 2, 'NORMAL', 3, 'HIGH'),  g.totmb_kfgrp, g.freemb_kfgrp,
g.minspc_kfgrp,  g.usable_kfgrp, g.offline_kfgrp,  g.compat_kfgrp,
g.dbcompat_kfgrp  from x$kfgrp_stat g where state_kfgrp != 0

GV$ASM_DISK_IOSTAT
select  d.inst_id, d.instname, d.dbname, d.group_number,  d.disk_number,
d.failgroup, d.reads,  d.writes, d.read_errs, d.write_errs, d.read_time,
d.write_time, d.bytes_read, d.bytes_written  from x$kfnsDskIost d


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$ASM_DISK_STAT
select  d.inst_id, d.grpnum_kfdsk, d.number_kfdsk,  d.compound_kfdsk,
d.incarn_kfdsk,  decode(d.mntsts_kfdsk,  1, 'MISSING', 2, 'CLOSED', 3,
'OPENED', 4, 'CACHED',  5, 'IGNORED', 6, 'IGNORED', 7, 'CLOSING', 8,
'IGNORED', 9, 'IGNORED',  'INVALID'),  decode(d.hdrsts_kfdsk,  1,
'UNKNOWN', 2, 'CANDIDATE', 3, 'MEMBER', 4, 'FORMER',  5, 'CONFLICT', 6,
'INCOMPATIBLE', 7, 'PROVISIONED', 8, 'FOREIGN',  'INVALID'),
decode(d.grpnum_kfdsk, 0, 'ONLINE', decode(d.mode_kfdsk,  0, 'UNKNOWN', 127,
'ONLINE', 21, 'OFFLINE', 1, 'OFFLINE', 'SYNCING')),  decode(d.state_kfdsk,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
1, 'UNKNOWN', 2, 'NORMAL', 3, 'UNUSED', 4, 'DROPPING',  5,
decode(d.mode_kfdsk, 127, 'DROPPING', 'FORCING'),  6, 'FORCING', 7, 'DROPPED',
8, 'ADDING', 'INVALID'),  decode(d.redun_kfdsk,  16, 'UNPROT', 17, 'UNPROT',
18, 'MIRROR', 19, 'MIRROR',  20, 'MIRROR', 21, 'MIRROR', 22, 'MIRROR', 23,
'MIRROR',  32, 'PARITY', 33, 'PARITY', 34, 'PARITY', 35, 'PARITY',  36,
'PARITY', 37, 'PARITY', 38, 'PARITY', 39, 'PARITY',  'UNKNOWN'),
d.libnam_kfdsk, d.osmb_kfdsk, d.totmb_kfdsk, d.totmb_kfdsk - d.usedmb_kfdsk,
d.asmname_kfdsk, d.failname_kfdsk, d.label_kfdsk, d.path_kfdsk,  d.udid_kfdsk,
d.product_kfdsk, d.crdate_kfdsk, d.mtdate_kfdsk,  d.timer_kfdsk,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
```

k.read_kfkid, k.write_kfkid, k.rerr_kfkid, k.werr_kfkid,
k.rtime_kfkid/1000000, k.wtime_kfkid/1000000, k.bytesr_kfkid, k.bytesw_kfkid,
decode(d.prefrd_kfdsk, 0, ' ', 1, 'Y', 2, 'N', 3, 'U') from x$kfdsk_stat d,
x$kfkid k where d.mntsts_kfdsk != 0 and d.kfkid_kfdsk = k.idptr_kfkid(+)

GV$ASM_FILE
select inst_id, group_kffil, number_kffil, compound_kffil, incarn_kffil,
blksiz_kffil, blkcnt_kffil, filsiz_kffil, filspc_kffil, sftype_kffil,
decode(redun_kffil, 17, 'UNPROT', 18, 'MIRROR', 19, 'HIGH', 35,

VIEW_NAME

VIEW_DEFINITION

'PARITY', 36, 'PARITY', 37, 'PARITY', 38, 'PARITY'),
decode(bitand(fdflg_kffil, 2), 2, 'FINE', 'COARSE'), crdate_kffil,
mddate_kffil, decode(thinned_kffil, 0, 'U', 4294967295, 'N', 'Y') from x$kffil
where incarn_kffil <> 0 and number_kffil > 255

GV$ASM_OPERATION
select inst_id, number_kfgbrb, 'REBAL', decode(error_kfgbrb, 0,
decode((select state_kfgmg from x$kfgmg where
number_kfgmg=number_kfgbrb and op_kfgmg <> 4), 1, 'WAIT', 2,

VIEW_NAME

VIEW_DEFINITION

'RUN', 3, 'REAP', 4, 'WAIT', 5, 'HALT', 6, 'ERRS', 'WAIT'),
'ERRS'), power_kfgbrb, (select actual_kfgmg from x$kfgmg where
number_kfgmg=number_kfgbrb and op_kfgmg <> 4), (select sofar_kfgmg from
x$kfgmg where number_kfgmg=number_kfgbrb and op_kfgmg <> 4), (select
work_kfgmg from x$kfgmg where number_kfgmg=number_kfgbrb and op_kfgmg <> 4),
(select rate_kfgmg from x$kfgmg where number_kfgmg=number_kfgbrb and op_kfgmg
<> 4), (select time_kfgmg from x$kfgmg where number_kfgmg=number_kfgbrb and
op_kfgmg <> 4), decode(error_kfgbrb, 0, '', 'ORA-' || error_kfgbrb) from
x$kfgbrb union all select inst_id, number_kfgmg, 'ONLIN', 'RUN', 1, 0, 0, 0,

VIEW_NAME

VIEW_DEFINITION

0, 0, '' from x$kfgmg where op_kfgmg = 4

GV$ASM_TEMPLATE
select inst_id, group_kftmta, entry_kftmta, decode(redundancy_kftmta, 17,
'UNPROT', 18, 'MIRROR', 19, 'HIGH', 35, 'PARITY', 36, 'PARITY', 37,
'PARITY', 38, 'PARITY'), decode(bitand(flags_kftmta, 1), 1, 'FINE', 'COARSE'),
decode(bitand(flags_kftmta, 4), 4, 'Y', 'N'), name_kftmta from x$kftmta

GV$AW_AGGREGATE_OP

VIEW_NAME

VIEW_DEFINITION
---
select inst_id, name_xsagopft as name, desc_xsagopft as longname, case when weight_xsagopft >= 0 then weight_xsagopft else null end as default_weight from x$xsagop where not bitand(flags_xsagopft, 32) = 0

GV$AW_ALLOCATE_OP
select inst_id, name_xsagopft as name, desc_xsagopft as longname from x$xsagop where not bitand(flags_xsagopft, 64) = 0

GV$AW_CALC

VIEW_NAME

VIEW_DEFINITION
---
select inst_id, session_id, agcachhit, agcachmiss, scachesuccess, scachefailure, pgcachhit, pgcachmiss, pgnewpage, pgscrounge, pgcachewrite, pgpoolsize, cdmlcmd, pdmlcmd, aggr_func_logical_na, aggr_func_precompute, aggr_func_calcs from x$xsaggr

GV$AW_LONGOPS
select inst_id, session_id, cursor_name, decode(command, 1, 'FETCH', 2, 'IMPORT', 3, 'EXECUTE', '?'), decode(status, 4, 'EXECUTING', 5, 'FETCHING', 6, 'FINISHED', '?'), rows_processed, start_time from x$xslongops

VIEW_NAME

VIEW_DEFINITION

GV$AW_OLAP
select a.inst_id, s.ksusenum, a.awnum, decode(a."MODE", 0, 'READ ONLY', 1, 'READ WRITE', 2, 'MULTIWRITE', 3, 'EXCLUSIVE', 'UNKNOWN'), a.gen_xsawso, a.temp_lob_count, a.temp_lob_read, a.perm_lob_read, a.changed_cache, a.unchanged_cache from x$ksuse s, x$xsawso a where s.addr = a.KSSOBOWN and bitand(a.flags, 128) = 0

GV$AW_SESSION_INFO

VIEW_NAME

VIEW_DEFINITION
---
select inst_id, session_id, client, state, sesshandle, userid, tottrns, trntime/1000000, tottrntime/1000000, (tottrntime/tottrns)/1000000, trncputime/1000000, tottrncputime/1000000, (tottrncputime/tottrns)/1000000 from x$xssinfo

GV$BACKUP

```
select inst_id,hxfil, decode(hxerr, 0,decode(bitand(fhsta, 1), 0,'NOT
ACTIVE','ACTIVE'), 1,'FILE MISSING', 2,'OFFLINE NORMAL', 3,'NOT VERIFIED',
4,'FILE NOT FOUND', 5,'CANNOT OPEN FILE', 6,'CANNOT READ HEADER', 7,'CORRUPT

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
HEADER', 8,'WRONG FILE TYPE', 9,'WRONG DATABASE', 10,'WRONG FILE NUMBER',
11,'WRONG FILE CREATE', 12,'WRONG FILE CREATE', 16,'DELAYED OPEN', 'UNKNOWN
ERROR'), to_number(fhbsc),  to_date(fhbti,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian') from x$kcvfhonl

GV$BACKUP_ASYNC_IO
select inst_id, sid, ser, setid, rman_status_recid, rman_status_stamp, devtype,
decode(type, 1, 'INPUT', 2, 'OUTPUT', 3, 'AGGREGATE', 'UNKNOWN'), decode(status,
1, 'NOT STARTED',              2, 'IN PROGRESS', 3, 'FINISHED', 'UNKNOWN'),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
filename, set_count, set_stamp, block_size * buffer_size, buffer_count,
decode(total_blocks, 0, null, total_blocks) * block_size, to_date(open_time,
'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'), to_date(close_time, 'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'), abs((to_date(close_time, 'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian') - to_date(open_time, 'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian')) * 8640000),decode(aggregate_count, 0,
null, aggregate_count) * 1, blocks * block_size,
decode(instr(open_time,close_time), 1, null, round((blocks * block_size) /
abs(((to_date(close_time, 'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian') -

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
to_date(open_time, 'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian')) * 86400))))
* 1, async_short_count + async_long_count + async_ready, async_ready,
async_short_count, async_short_tottime, async_short_maxtime, async_long_count,
async_long_tottime, async_long_maxtime from x$ksfqp where bitand(flags,2) = 2

GV$BACKUP_CORRUPTION
select
inst_id,fcrid,fcstm,fcbss,fcbsc,fcpno,fcdfp,fcblk,fccnt,to_number(fcscn),decode(
bitand(fcflg,1),1,'YES','NO'),decode(bitand(fcflg,30),2,'ALL

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
ZERO',4,'FRACTURED',8,'CHECKSUM',                  16,'CORRUPT',
decode(to_number(fcscn),0,'UNKNOWN','LOGICAL'))  from x$kccfc
```

```
GV$BACKUP_DATAFILE
select
inst_id,bfrid,bfstm,bfbss,bfbsc,bfdfp,to_number(bfcrs),to_date(bfcrt,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(bfrls),to_date(bfrlc,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(bitand(bfflg,1+8),1,bflvl,NULL),to_
number(bfics),to_number(bfcps),to_date(bfcpt,'MM/DD/RR

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(bfafs),bfncb,bfmcb,bflcb,bffsz,b
fbct,bfbsz,bflor,to_date(bftsm, 'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(bfdfp, 0,
decode(bitand(bfflg,2),2,'S','B'), NULL),
decode(bitand(bfflg,4),4,'YES','NO'),bfbrd,
decode(bitand(bfflg,16),16,'YES','NO'), bffdi,
decode(bitand(bfflg,32),0,'NO','YES'), bfplus, bfprls, bfprlt, bfssz,
decode(bitand(bfflg,128),0,'NO','YES') from x$kccbf where bitand(bfflg, 64) !=
64

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

GV$BACKUP_DEVICE
select inst_id, devtype, devname from x$ksfqdvnt

GV$BACKUP_PIECE
select inst_id,bprid,bpstm,bpbss,bpbsc,bpnum,bitand(bpflg, 12)/4 +
(bitand(bpext, 64-1) * 4) + 1,bpdev,bphdl,bpcmt,bpmdh,bitand(bpflg, 4080) /
16,decode(bitand(bpflg,2),1,'YES','NO'),bptag,decode(bitand(bpflg,
1+4096+8192),0,'A',1,'D',4096,'X',8192,'U','?'),decode(bitand(bpflg,1),1,'YES','

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
NO'),to_date(bptsm,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_date(bptim,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),abs((to_date(bptim,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian')- to_date(bptsm,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'))*86400),((floor(bpext/512) * 4294967296) +
bpsz1) * 512, decode(bitand(bpflg, 16384),0,'NO','YES'),
bprsi,bprst,decode(bitand(bpext, 64),64,'YES','NO'), decode(bitand(bpflg,
16384),0,'NO',decode(bitand(bpext, 256),0,'NO','YES')), decode(bitand(bpext,
128),128,'YES','NO'), decode(bitand(bpflg,

VIEW_NAME
------------------------------
```

```
VIEW_DEFINITION
--------------------------------------------------------------------------------
16384),16384,'NO',decode(bitand(bpext,256),0,'NO','YES')) from x$kccbp

GV$BACKUP_REDOLOG
select
inst_id,blrid,blstm,blbss,blbsc,blthp,blseq,to_number(blrls),to_date(blrlc,'MM/D
D/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(bllos),to_date(bllot,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(blnxs),to_date(blnxt,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),blbct,blbsz, decode(bitand(blflg, 1), 1,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
'YES', 'NO') from x$kccbl where bitand(blflg, 2) != 2

GV$BACKUP_SET
select
inst_id,bsrid,bsstm,bsbss,bsbsc,decode(bitand(bstyp,11),1,'D',2,'I',8,'L'),decod
e(bitand(bstyp,4+64),4,'YES',68,'SBY','NO'),decode(bitand(bstyp,16+8192),16,bslv
l,NULL),bspct,to_date(bsbst,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(bitand(bstyp, 4096),4096,
to_date(bsbst,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
to_date(bstsm,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian')),decode(bitand(bstyp, 4096),4096,0,
abs((to_date(bstsm,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian')-
to_date(bsbst,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'))*86400)),bsbsz,
decode(bitand(bstyp,128),128,'YES','NO'),  decode(bitand(bstyp, 1792), 0, 'NO',
'YES'),         to_date(bskpt,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(bitand(bstyp, 1792), 256,  'LOGS',
512,  'NOLOGS',                          1024, 'BACKUP_LOGS',
NULL),         decode(bitand(bstyp, 16384), 16384, 'YES', 'NO') from x$kccbs

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
where bitand(bstyp,32) != 32

GV$BACKUP_SPFILE
select inst_id,birid,bistm,bibss,bibsc,  to_date(bimdt, 'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian') , bifsz, to_date(bitsm, 'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'), bidun  from x$kccbi  where bitand(biflg,
1) != 1
```

GV$BACKUP_SYNC_IO

VIEW_NAME

VIEW_DEFINITION
---
select inst_id, sid, ser, setid, rman_status_recid, rman_status_stamp, devtype,
decode(type, 1, 'INPUT', 2, 'OUTPUT', 3, 'AGGREGATE', 'UNKNOWN'), decode(status,
1, 'NOT STARTED', 2, 'IN PROGRESS', 3, 'FINISHED', 'UNKNOWN'),
filename, set_count, set_stamp, block_size * buffer_size, buffer_count,
decode(total_blocks, 0, null, total_blocks) * block_size, to_date(open_time,
'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'), to_date(close_time, 'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'), abs((to_date(close_time, 'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian') - to_date(open_time, 'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian')) * 8640000),decode(aggregate_count, 0,

VIEW_NAME

VIEW_DEFINITION
---
null, aggregate_count) * 1, blocks * block_size,
decode(instr(open_time,close_time), 1, null, round((blocks * block_size) /
abs(((to_date(close_time, 'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian') -
to_date(open_time, 'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian')) * 86400))))
* 1, sync_count, sync_tottime, sync_maxtime, decode(sync_tottime, 0, NULL,
round((blocks * block_size) / sync_tottime * 100)) * 1 from x$ksfqp where
bitand(flags,2) = 0

GV$BGPROCESS

VIEW_NAME

VIEW_DEFINITION
---
select p.inst_id, p.ksbdppro,p.ksbdpser,p.ksbdpnam,d.ksbdddsc,p.ksbdperr from
x$ksbdp p,x$ksbdd d where p.indx=d.indx and p.ksbdpnam not like 'TEST%'

GV$BH
select bh.inst_id, file#, dbablk, class,
decode(state,0,'free',1,'xcur',2,'scur',3,'cr',
4,'read',5,'mrec',6,'irec',7,'write',8,'pi',
9,'memory',10,'mwrite',11,'donated', 12,'protected', 13,'securefile',
14,'siop',15,'recckpt'), 0, 0, 0, bh.le_addr, name,le_class,

VIEW_NAME

VIEW_DEFINITION
---
decode(bitand(flag,1), 0, 'N', 'Y'), decode(bitand(flag,16), 0, 'N', 'Y'),
decode(bitand(flag,1536), 0, 'N', 'Y'), decode(bitand(flag,16384), 0, 'N', 'Y'),
decode(bitand(flag,65536), 0, 'N', 'Y'), 'N', obj, ts#, lobid from x$bh bh,
x$le le where bh.le_addr = le.le_addr (+)

GV$BLOCKING_QUIESCE
select inst_id, sid_kgskvft from x$kgskvft where active_kgskvft = 1
and mapped_cg_name_kgskvft <> 'SYS_GROUP'

VIEW_NAME

VIEW_DEFINITION

GV$BSP
select inst_id, reqcr, reqcur, reqdata, requndo, reqtx, rescur,
respriv, reszero, resdisk, resfail, fairdc, faircl, 0, flush, 0,
flushf, flushmx, light, signal from x$kclcrst

GV$BUFFERED_PUBLISHERS
select inst_id, queue_id, queue_schema, queue_name, sender_name,
sender_address, sender_protocol, num_msgs, cnum_msgs, last_enqueued_msg,
unbrowsed_msgs, overspilled_msgs, memory_usage, decode(bitand(publisher_flags,

VIEW_NAME

VIEW_DEFINITION
---
7), 1, 'IN FLOW CONTROL: TOO MANY UNBROWSED MESSAGES', 2, 'IN FLOW CONTROL:
OVERSPILLED MESSAGES', 4, 'IN FLOW CONTROL: INSUFFICIENT MEMORY AND UNBROWSED
MESSAGES', 0, 'PUBLISHING MESSAGES') from x$buffered_publishers

GV$BUFFERED_QUEUES
select inst_id, queue_id, queue_schema, queue_name, startup_time, num_msgs,
spill_msgs, cnum_msgs, cspill_msgs, expired_msgs from x$buffered_queues where
bitand(flags, 16) = 0

VIEW_NAME

VIEW_DEFINITION

GV$BUFFERED_SUBSCRIBERS
select s.inst_id, s.queue_id, q.queue_schema, q.queue_name, s.subscriber_id,
s.subscriber_name, s.subscriber_address, s.protocol, s.subscriber_type,
q.startup_time, s.last_browsed_seq, s.last_browsed_num, s.last_dequeued_seq,
s.last_dequeued_num, s.current_enq_seq, s.num_msgs, s.cnum_msgs,
s.total_dequeued_msg, s.total_spilled_msg, s.expired_msgs, s.message_lag from
x$buffered_subscribers s, x$buffered_queues q where s.inst_id = q.inst_id and
s.queue_id = q.queue_id and bitand(q.flags, 16) = 0

VIEW_NAME

VIEW_DEFINITION

GV$BUFFER_POOL

```
select  inst_id,                        bp_id,
bp_name,                        bp_blksz,
decode(bp_state, 0, 'STATIC',     1, 'ALLOCATING',
2, 'ACTIVATING', 3, 'SHRINKING'),           bp_currgrans * bp_gransz,
bp_size,                        bp_tgtgrans  * bp_gransz,
bp_tgtgrans  * bp_bufpergran,         bp_prevgrans * bp_gransz,
bp_prevgrans * bp_bufpergran,         0, 0,
bp_lo_sid,                      bp_hi_sid,
```

VIEW_NAME

VIEW_DEFINITION

```
bp_set_ct                 from x$kcbwbpd where bp_id > 0 and bp_currgrans >
0                and bp_tgtgrans > 0
```

GV$BUFFER_POOL_STATISTICS

```
select kcbwbpd.inst_id, kcbwbpd.bp_id, kcbwbpd.bp_name, kcbwbpd.bp_blksz,
sum(kcbwds.cnum_set),sum(kcbwds.cnum_repl), sum(kcbwds.cnum_write),
sum(kcbwds.cnum_set), sum(kcbwds.buf_got), sum(kcbwds.sum_wrt),
sum(kcbwds.sum_scn), sum(kcbwds.fbwait), sum(kcbwds.wcwait), sum(kcbwds.bbwait),
sum(kcbwds.fbinsp), sum(kcbwds.dbinsp), sum(kcbwds.dbbchg), sum(kcbwds.dbbget),
```

VIEW_NAME

VIEW_DEFINITION

```
sum(kcbwds.conget), sum(kcbwds.pread), sum(kcbwds.pwrite) from x$kcbwds kcbwds,
x$kcbwbpd kcbwbpd where kcbwds.set_id >= kcbwbpd.bp_lo_sid and kcbwds.set_id <=
kcbwbpd.bp_hi_sid and kcbwbpd.bp_size != 0 group by kcbwbpd.inst_id,
kcbwbpd.bp_id, kcbwbpd.bp_name, kcbwbpd.bp_blksz
```

GV$CALLTAG

```
select w.inst_id, w.kywmnfnum, w.kywmnfpc, w.kywmnfwrc, w.kywmnfhct,
s.service_name, s.module, s.action, s.username, s.program        from x$kywmnf
w, v$session s where w.kywmnfnum = s.sid
```

VIEW_NAME

VIEW_DEFINITION

GV$CIRCUIT

```
select inst_id,kmcvcadr,kmcvcdpc,decode(kmcvcpro,kmcvcdpc,
hextoraw('00'),kmcvcpro),kmcvcwat,kmcvcses,kmcvcsta,kmcvcque,kmcvcsz0,
kmcvcsz1,kmcvcsz2,kmcvcsz3,kmcvcnmg,kmcvcnmb,kmcvcbrk,kmcvcpre,kmcvcpvc from
x$kmcvc where bitand(ksspaflg,1) != 0
```

GV$CLASS_CACHE_TRANSFER

```
select 0, 'data block', 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  from dual
```

VIEW_NAME

VIEW_DEFINITION

GV$CLASS_PING
select 0, 'data block', 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 from
dual

GV$CLIENT_RESULT_CACHE_STATS
select INST_ID, KPOQSTA_CLREGID, KPOQSTA_BSZ,
KPOQSTA_BMX, KPOQSTA_BCT, KPOQSTA_BUC, KPOQSTA_NEW,
KPOQSTA_FAI, KPOQSTA_FND, KPOQSTA_INV, KPOQSTA_DIV,

VIEW_NAME

VIEW_DEFINITION

KPOQSTA_DVA from x$kpoqsta

GV$CLIENT_STATS
select c.inst_id, c.clsnam, m.extid, m.sname, c.statval from x$kewecls c,
x$kewssmap m where c.clspos = m.offst and m.aggid = 5

GV$CLUSTER_INTERCONNECTS
SELECT INST_ID, NAME_SKGXPIA, IP_SKGXPIA, decode(PUB_SKGXPIA, 'Y',
'YES', 'N', 'NO'), decode(PICKED_SKGXPIA, 'OSD', 'OS dependent

VIEW_NAME

VIEW_DEFINITION
---
software', 'OCR', 'Oracle Cluster Repository',
'CI', 'cluster_interconnects parameter') FROM X$SKGXPIA

GV$CONFIGURED_INTERCONNECTS
SELECT INST_ID, NAME_KSXPIA, IP_KSXPIA, decode(PUB_KSXPIA, 'Y', 'YES',
'N', 'NO'), decode(PICKED_KSXPIA, 'OSD', 'OS dependent software',
'OCR', 'Oracle Cluster Repository', 'CI', 'cluster_interconnects
parameter') FROM X$KSXPIA

VIEW_NAME

VIEW_DEFINITION

GV$CONTEXT
select namespace, attribute, value from x$context

GV$CONTROLFILE
select inst_id,decode(bitand(cfflg,1),0,'',1,'INVALID'),cfnam,
decode(bitand(cffl2,1),0,'NO','YES'), cfbsz, cffsz from x$kcccf

GV$CONTROLFILE_RECORD_SECTION
select inst_id,decode(indx,0,'DATABASE',1, 'CKPT PROGRESS', 2, 'REDO

VIEW_NAME

VIEW_DEFINITION

THREAD',3,'REDO LOG',4,'DATAFILE',5,'FILENAME',6,'TABLESPACE',7,'TEMPORARY
FILENAME',8,'RMAN CONFIGURATION',9,'LOG HISTORY',10,'OFFLINE RANGE',11,'ARCHIVED
LOG',12,'BACKUP SET',13,'BACKUP PIECE',14,'BACKUP DATAFILE',15, 'BACKUP
REDOLOG',16,'DATAFILE COPY',17,'BACKUP CORRUPTION',18,'COPY
CORRUPTION',19,'DELETED OBJECT',20,'PROXY COPY',21,'BACKUP SPFILE',23,'DATABASE
INCARNATION',24,'FLASHBACK LOG',25, 'RECOVERY DESTINATION', 26,'INSTANCE SPACE
RESERVATION', 27, 'REMOVABLE RECOVERY FILES', 28, 'RMAN STATUS', 29, 'THREAD
INSTANCE NAME MAPPING', 30, 'MTTR', 31, 'DATAFILE HISTORY', 32, 'STANDBY
DATABASE MATRIX', 33, 'GUARANTEED RESTORE POINT', 34, 'RESTORE POINT', 35,

VIEW_NAME

VIEW_DEFINITION

'DATABASE BLOCK CORRUPTION', 36, 'ACM OPERATION', 37, 'FOREIGN ARCHIVED LOG',
'UNKNOWN'),rsrsz,rsnum,rsnus,rsiol,rsilw,rsrlw from x$kccrs where indx not in
(22)

GV$COPY_CORRUPTION
select
inst_id,ccrid,ccstm,ccdcp,ccdcs,ccdfp,ccblk,cccnt,to_number(ccscn),decode(bitand
(ccflg,1),1,'YES','NO'),decode(bitand(ccflg,30),2,'ALL
ZERO',4,'FRACTURED',8,'CHECKSUM', 16,'CORRUPT',

VIEW_NAME

VIEW_DEFINITION
---
decode(to_number(ccscn),0,'UNKNOWN','LOGICAL')) from x$kcccc

GV$CORRUPT_XID_LIST
select INST_ID, CORRUPT_XID from x$ktucus

GV$CPOOL_CC_INFO
select POOL_NAME, CCLASS_NAME, INST_ID from x$kpplcc_info

GV$CPOOL_CC_STATS

VIEW_NAME

VIEW_DEFINITION
---
select CCLASS_NAME, INST_ID, NUM_REQUESTS, NUM_HITS,
NUM_MISSES, NUM_WAITS, WAIT_TIME, CLIENT_REQ_TIMEOUTS,

NUM_AUTHENTICATIONS from x$kpplcc_stats

GV$CPOOL_STATS
select POOL_NAME, INST_ID, NUM_OPEN_SERVERS,
NUM_BUSY_SERVERS, NUM_AUTH_SERVERS, NUM_REQUESTS, NUM_HITS,
NUM_MISSES, NUM_WAITS, WAIT_TIME, CLIENT_REQ_TIMEOUTS,
NUM_AUTHENTICATIONS, NUM_PURGED, HISTORIC_MAX from x$kpplcp_stats

VIEW_NAME

VIEW_DEFINITION

GV$CR_BLOCK_SERVER
select inst_id, reqcr, reqcur, reqdata, requndo, reqtx, reqother,
rescur, respriv, reszero, resdisk, resfail, stale, fairdc, faircl, 0,
flush, 0, flushf, flushmx, light, signal
from x$kclcrst

GV$CURRENT_BLOCK_SERVER
select inst_id, pin1, pin10, pin100, pin1000, pin10000, flush1,

VIEW_NAME

VIEW_DEFINITION
---
flush10, flush100, flush1000, flush10000, write1, write10,
write100, write1000, write10000, cleandc, rcvdc, queuedc,
evictdc, writedc from x$kclcurst

GV$DATABASE
select di.inst_id,di.didbi,di.didbn,to_date(di.dicts,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(di.dirls),to_date(di.dirlc,'MM/D
D/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(di.diprs),to_date(di.diprc,'MM/D

VIEW_NAME

VIEW_DEFINITION

D/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(di.dimla,0,'NOARCHIVELOG',1,'ARCHIV
ELOG','MANUAL'),to_number(di.discn),to_number(di.difas),decode(bitand(di.diflg,2
56),256,'CREATED',decode(bitand(di.diflg,1024),1024,'STANDBY',decode(bitand(di.d
iflg,32768),32768,'CLONE',decode(bitand(di.diflg,4096),4096,'BACKUP','CURRENT'))
)),to_date(di.dicct,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),di.dicsq,to_number(di.dickp_scn),to_date(d
i.dickp_tim,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(bitand(di.diflg,4),4,'REQUIRED',dec

VIEW_NAME

```
VIEW_DEFINITION
--------------------------------------------------------------------------------
ode(di.diirs,0,'NOT ALLOWED','ALLOWED')),to_date(di.divts,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(di.didor,0,'MOUNTED',decode(di.dido
r,1,'READ WRITE','READ ONLY')),decode(bitand(di.diflg,65536),65536,'MAXIMUM
PROTECTION',decode(bitand(di.diflg,128),128,'MAXIMUM
AVAILABILITY',decode(bitand(di.diflg,134217728),134217728,'RESYNCHRONIZATION',de
code(bitand(di.diflg,8),8,'UNPROTECTED','MAXIMUM
PERFORMANCE'))))),decode(di.diprt,1,'MAXIMUM PROTECTION',2,'MAXIMUM
AVAILABILITY',3,'RESYNCHRONIZATION',4,'MAXIMUM PERFORMANCE',5,'UNPROTECTED',
'UNKNOWN'),decode(di.dirae,0,'DISABLED',1,'SEND',2,'RECEIVE',3,'ENABLED','UNKNOW

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
N'),to_number(di.diacid),to_number(di.diacid),decode(bitand(di.difl2,32768),3276
8,'SNAPSHOT STANDBY',decode(bitand(di.diflg,33554432),33554432,'LOGICAL
STANDBY',decode(bitand(di.diflg,1024),1024,'PHYSICAL
STANDBY','PRIMARY'))),to_number(di.diars),decode(bitand(difl2,1),1,'ENABLED','DI
SABLED'),decode(di.disos,0,'IMPOSSIBLE',1,'NOT ALLOWED',2,'SWITCHOVER
LATENT',3,'SWITCHOVER PENDING',4,'TO PRIMARY',5,'TO STANDBY',6,'RECOVERY
NEEDED',7,'SESSIONS ACTIVE',8,'PREPARING SWITCHOVER',9,'PREPARING
DICTIONARY',10,'TO LOGICAL
STANDBY','UNKNOWN'),decode(di.didgd,0,'DISABLED','ENABLED'),decode(bitand(di.dif

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
lg,1048576),1048576,'ALL',decode(bitand(di.diflg,2097152),2097152,'STANDBY','NON
E')),decode(bitand(diflg,1073741824),1073741824, 'YES',
decode(bitand(diflg, 131072 + 262144 + 524288),0,
decode(bitand(difl2,2 + 64), 0,'NO','IMPLICIT'),
'IMPLICIT')),decode(bitand(di.diflg,131072),131072,'YES','NO'),decode(bitand(di.
diflg,262144),262144,'YES','NO'),decode(bitand(di.diflg,268435456),268435456,'YE
S','NO'),di.diplid, di.dipln, di2.di2rdi,
di2.di2inc,to_number(di.dicur_scn),decode(bitand(di2.di2flag,1),1,'YES',
decode(di2.di2rsp_oldest,0,'NO','RESTORE POINT

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
ONLY')),decode(bitand(diflg,524288),524288,'YES','NO'),decode(bitand(difl2,2),2,
'YES','NO'),di2.di2dbun, to_number(di2.di2actiscn),
decode(di.difsts,0,'DISABLED',1,'BYSTANDER',2,'SYNCHRONIZED',3,'UNSYNCHRONIZED',
4,'SUSPENDED',5,'STALLED',6,'LOADING DICTIONARY',7,'PRIMARY
UNOBSERVED',8,'REINSTATE REQUIRED',9,'REINSTATE IN PROGRESS',10,'REINSTATE
FAILED',11,'TARGET OVER LAG LIMIT',12,'TARGET UNDER LAG LIMIT',''), di.diftgt,
di.difths,decode(di.difopr,1,'YES',2,'NO',3,'UNKNOWN',''), di.difobs,
decode(bitand(difl2, 16384), 16384, 'YES', 'NO'), di2.di2pdbun,
```

```
decode(bitand(di.difl2,64), 64, 'YES', 'NO'), decode(di2.di2min_req_capture_scn,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
0, to_number(null),        di2.di2min_req_capture_scn) from x$kccdi di,
x$kccdi2 di2

GV$DATABASE_BLOCK_CORRUPTION
select inst_id, cor.blkfno, cor.blksblk, cor.blktot, cor.blkscn,
decode(cor.blktype, 2, 'ALL ZERO', 3,'FRACTURED', 4,'CHECKSUM',
5,'CORRUPT', decode(cor.blkscn, 0, 'UNKNOWN', 'LOGICAL'))  from x$kccblkcor cor,
v$datafile df   where cor.blktype != 1    and cor.blkfno = df.file#    and
cor.blkcrs = df.creation_change#    and cor.blkcrt = df.creation_time

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

GV$DATABASE_INCARNATION
select userenv('Instance'),  icrid, to_number(icrls),
to_date(icrlc,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),
to_number(icprs),        to_date(icprc,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),        decode(bitand(icflg,3),
1,'ORPHAN', 2, 'CURRENT', 0, 'PARENT',              'ORPHAN'),
icrlc_i, icpinc, icalw  from x$kccic


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$DATAFILE
select /*+ rule */ fe.inst_id,fe.fenum,to_number(fe.fecrc_scn),
to_date(fe.fecrc_tim,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),
fe.fetsn,fe.ferfn,
decode(fe.fetsn,0,decode(bitand(fe.festa,2),0,'SYSOFF','SYSTEM'),
decode(bitand(fe.festa,18),0,'OFFLINE',2,'ONLINE','RECOVER')),
decode(fe.fedor,2,'READ ONLY',   decode(bitand(fe.festa, 12),
0,'DISABLED',4,'READ ONLY',12,'READ WRITE','UNKNOWN')), to_number(fe.fecps),
to_date(fe.fecpt,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
to_number(fe.feurs), to_date(fe.feurt,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'), to_number(fe.fests),
decode(fe.fests,NULL,to_date(NULL),  to_date(fe.festt,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian')),
```

```
to_number(fe.feofs),to_number(fe.feonc_scn), to_date(fe.feonc_tim,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),
fh.fhfsz*fe.febsz,fh.fhfsz,fe.fecsz*fe.febsz,fe.febsz,fn.fnnam, fe.fefdb,
fn.fnbof,  decode(fe.fepax, 0, 'UNKNOWN', 65535, 'NONE', fnaux.fnnam),
to_number(fh.fhfirstunrecscn),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
to_date(fh.fhfirstunrectime,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),
fe.fepdi, fe.fefcrs, fe.fefcrt, decode(fe.fefdb, 1, 'YES', 'NO'),  fe.feplus,
fe.feprls, fe.feprlt  from x$kccfe fe, x$kccfn fn, x$kccfn fnaux, x$kcvfh fh
where
((fe.fepax!=65535 and fe.fepax!=0 and fe.fepax=fnaux.fnnum)           or
((fe.fepax=65535 or fe.fepax=0) and fe.fenum=fnaux.fnfno
and fnaux.fntyp=4 and fnaux.fnnam is not null and
bitand(fnaux.fnflg, 4) != 4                                    and
fe.fefnh=fnaux.fnnum))                                  and

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
fn.fnfno=fe.fenum and fn.fnfno=fh.hxfil and fe.fefnh=fn.fnnum          and
fe.fedup!=0 and fn.fntyp=4 and fn.fnnam is not null  and bitand(fn.fnflg, 4) !=
4

GV$DATAFILE_COPY
select
inst_id,dcrid,dcstm,dcnam,dctag,dcdfp,dcrfn,to_number(dccrs),to_date(dccrt,'MM/D
D/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(dcrls),to_date(dcrlc,'MM/DD/RR

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(bitand(dcflg,8),8,0,NULL),to_number
(dccps),to_date(dccpt,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(dcafs),to_number(dcrfs),to_date(
dcrft,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(bitand(dcflg,
2),0,'NO','YES'),decode(bitand(dcflg,
4),0,'NO','YES'),dcncb,dcmcb,dclcb,dcbct,dcbsz,dclor,decode(bitand(dcflg,
1),0,'NO','YES'),decode(bitand(dcflg, 1+32+64+4096),0,'A',1,'D',32,'X',64,'U',
4096+1,'F','?'),to_date(dctsm,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),
decode(dcdfp, 0, decode(bitand(dcflg, 16),16,'S','B'),NULL),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
```

```
decode(bitand(dcflg, 1792), 0, 'NO',
'YES'),          to_date(dckpt,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(bitand(dcflg, 1792), 256,  'LOGS',
512,  'NOLOGS',                  1024, 'BACKUP_LOGS',
NULL),        decode(bitand(dcflg, 128),0,'NO','YES'), decode(bitand(dcflg,
2048),0,'NO','YES'), dcrsi, dcrst, decode(bitand(dcflg, 4096),0,'NO','YES'),
decode(bitand(dcflg, 4096+8192),0,NULL,4096,'NO','YES'), dcfdi,
decode(bitand(dcflg, 16384), 0, 'NO', 'YES'), dcplus, dcprls, dcprlt,
decode(bitand(dcflg, 32768), 0, 'NO', 'YES') from x$kccdc

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

GV$DATAFILE_HEADER
select inst_id,hxfil,decode(hxons, 0, 'OFFLINE', 'ONLINE'),decode(hxerr, 0,
NULL, 1,'FILE MISSING',2,'OFFLINE NORMAL', 3,'NOT VERIFIED', 4,'FILE NOT
FOUND',5,'CANNOT OPEN FILE', 6,'CANNOT READ HEADER', 7,'CORRUPT HEADER',8,'WRONG
FILE TYPE', 9,'WRONG DATABASE', 10,'WRONG FILE NUMBER',11,'WRONG FILE CREATE',
12,'WRONG FILE CREATE', 16,'DELAYED OPEN',14, 'WRONG RESETLOGS', 15,'OLD
CONTROLFILE', 'UNKNOWN ERROR'),hxver,decode(hxnrcv, 0,'NO', 1,'YES',
NULL),decode(hxifz, 0,'NO', 1,'YES',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
NULL),to_number(fhcrs),to_date(fhcrt,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),fhtnm,fhtsn,fhrfn,to_number(fhrls),to_date
(fhrlc,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(fhscn),to_date(fhtim,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),fhcpc,fhfsz*fhbsz,fhfsz,hxfnm,
decode(hxlmdba, 0, NULL, hxlmdba), decode(hxlmld_scn, to_number('0'), NULL,
hxlmld_scn), decode(hxuopc_scn, 0, NULL, hxuopc_scn) from x$kcvfh

GV$DATAGUARD_CONFIG

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select DGCDBUN from x$krstdgc

GV$DATAGUARD_STATUS
select inst_id, decode(agfac,1,'Crash Recovery',          2,'Log Transport
Services',          3,'Log Apply Services',          4,'Role Management
Services',          5,'Remote File Server',          6,'Fetch Archive
Log',          7,'Data Guard',          8,'Network Services',
'UNKNOWN'), decode(agsev,1,'Informational',          2,'Warning',
3,'Error',          4,'Fatal',          5,'Control',
```

VIEW_NAME

VIEW_DEFINITION

'UNKNOWN'), agdid, agseq, agoer, decode(bitand(agflg, 1),0,'NO','YES'),
to_date(agdat,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'), agtxt from
x$krstalg order by agseq

GV$DATAPUMP_JOB
SELECT inst_id, kupvjid, kupvjjob, kupvjowner, kupvjctrlque,
kupvjstatque, kupvjoperation, kupvjmode, kupvjmasterid,
kupvjstate, kupvjworkers, kupvjflags, kupvjserialnum
FROM x$kupvj

VIEW_NAME

VIEW_DEFINITION

GV$DATAPUMP_SESSION
SELECT inst_id, kupvaid, kupvajobid, kupvasesaddr,
decode(kupvasestype,1,'DBMS_DATAPUMP',2,'MASTER',3,'WORKER',
4,'EXTERNAL TABLE','OTHER') FROM x$kupva

GV$DBFILE
select inst_id,fnfno,fnnam from x$kccfn where fnnam is not null and
bitand(fnflg, 4) != 4 and fntyp=4

VIEW_NAME

VIEW_DEFINITION

GV$DBLINK
select inst_id,nconam, ncouid, decode(bitand(hstflg, 32), 0, 'NO', 'YES'),
decode(bitand(hstflg, 8), 0, 'NO', 'YES'), decode(hstpro, 1, 'V5', 2, 'V6', 3,
'V6_NLS', 4, 'V7', 'UNKN'), ncouct, decode(bitand(ncoflg, 2), 0, 'NO', 'YES'),
decode(bitand(ncoflg, 8), 0, 'NO', 'YES'), nco2pstr from x$uganco where
bitand(hstflg, 1) != 0

GV$DB_CACHE_ADVICE

VIEW_NAME

VIEW_DEFINITION
---
select A.inst_id, A.bpid, B.bp_name, A.blksz, decode
(A.status, 2, 'ON', 'OFF'), A.poolsz,
round((A.poolsz / A.actual_poolsz), 4), A.nbufs,
decode (A.base_preads, 0, to_number(null),
round((A.preads / A.base_preads), 4)), decode
(A.base_preads, 0, A.actual_preads,

```
round((A.preads * (A.actual_preads / A.base_preads)), 0)),
A.estd_time_for_disk_reads,                              decode
(A.total_db_time, 0, A.estd_time_for_disk_reads,                round((100 *

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
A.estd_time_for_disk_reads / A.total_db_time), 1)), A.estd_rac_reads,
A.estd_rac_time                                          from x$kcbsc
A, x$kcbwbpd B                                 where A.bpid = B.bp_id and
A.inst_id = B.inst_id                order by A.inst_id, A.bpid, A.poolsz

GV$DB_OBJECT_CACHE
select inst_id,kglnaown,kglnaobj,kglnadlk,
decode(kglhdnsp,0,'CURSOR',1,'TABLE/PROCEDURE',2,'BODY',3,'TRIGGER',
4,'INDEX',5,'CLUSTER',6,'OBJECT',13,'JAVA SOURCE',14,'JAVA RESOURCE',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
15,'REPLICATED TABLE OBJECT',16,'REPLICATION INTERNAL PACKAGE', 17,'CONTEXT
POLICY',18,'PUB_SUB',19,'SUMMARY',20,'DIMENSION', 21,'APP CONTEXT',22,'STORED
OUTLINE',23,'RULESET',24,'RSRC PLAN', 25,'XML SCHEMA',26,'PENDING RSRC
PLAN',27,'PENDING RSRC CONSUMER GROUP',
28,'SUBSCRIPTION',29,'LOCATION',30,'REMOTE OBJECT',  31,'SNAPSHOT
METADATA',32,'JAVA SHARED DATA',33,'SECURITY PROFILE',  43, 'XDB CONFIG', 63,
'XDB ACL', 'INVALID NAMESPACE'), decode(bitand(kglobflg,3),0,'NOT
LOADED',2,'NON-EXISTENT',3,'INVALID STATUS', decode(kglobtyp,
0,'CURSOR',1,'INDEX',2,'TABLE',3,'CLUSTER',4,'VIEW',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
5,'SYNONYM',6,'SEQUENCE',7,'PROCEDURE',8,'FUNCTION',9,'PACKAGE',10,
'NON-EXISTENT',11,'PACKAGE BODY',12,'TRIGGER',13,'TYPE',14,'TYPE BODY',
15,'OBJECT',16,'USER',17,'DBLINK',18,'PIPE',19,'TABLE PARTITION', 20,'INDEX
PARTITION',21,'LOB',22,'LIBRARY',23,'DIRECTORY',24,'QUEUE', 25,'INDEX-ORGANIZED
TABLE',26,'REPLICATION OBJECT GROUP', 27,'REPLICATION PROPAGATOR', 28,'JAVA
SOURCE',29,'JAVA CLASS',30,'JAVA RESOURCE',31,'JAVA JAR', 32,'INDEX TYPE',33,
'OPERATOR',34,'TABLE SUBPARTITION',35,'INDEX SUBPARTITION', 36, 'REPLICATED
TABLE OBJECT',37,'REPLICATION INTERNAL PACKAGE', 38,'CONTEXT
POLICY',39,'PUB_SUB',40,'LOB PARTITION',41,'LOB SUBPARTITION',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
42,'SUMMARY',43,'DIMENSION',44,'APP CONTEXT',45,'STORED OUTLINE',46,'RULESET',
47,'RSRC PLAN',48,'RSRC CONSUMER GROUP',49,'PENDING RSRC PLAN', 50,'PENDING RSRC
```

```
CONSUMER GROUP',51,'SUBSCRIPTION',52,'LOCATION', 53,'REMOTE OBJECT',54,'SNAPSHOT
METADATA',55,'XDB', 56,'JAVA SHARED DATA',57,'SECURITY PROFILE','INVALID
TYPE')), kglobhs0+kglobhs1+kglobhs2+kglobhs3+kglobhs4+kglobhs5+kglobhs6,
kglhdldc,kglhdexc,kglhdlkc,kglobpc0,decode(kglhdkmk,0,'NO','YES'),kglhdclt,
kglhdivc from x$kglob

GV$DB_PIPES

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select inst_id,decode(kglobt00,1,kglobt17,null),kglnaobj,
decode(kglobt00,1,'PRIVATE','PUBLIC'),
kglobhs0+kglobhs1+kglobhs2+kglobhs3+kglobhs4+kglobhs5+kglobhs6 from x$kglob
where kglhdnsp=7 and kglobsta != 0

GV$DB_TRANSPORTABLE_PLATFORM
select INST_ID, PLATFORM_ID, PLATFORM_NAME,         decode(endian_format,
1,'Big' ,0,'Little','UNKNOWN FORMAT')         from x$kcpxpl         where
endian_format =          (select endian_format from x$kcpxpl pl, x$kccdi di

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
where pl.platform_id = di.diplid)

GV$DELETED_OBJECT
select inst_id,dlrid,dlstm,decode(dltyp,11,'ARCHIVED LOG',13,'BACKUP
PIECE',16,'DATAFILE COPY',20,'PROXY COPY',34,'RESTORE POINT',35,'DATABASE BLOCK
CORRUPTION', 37,'FOREIGN ARCHIVED LOG', 255,'BACKUP PIECE AVAILABLE',254,'BACKUP
PIECE EXPIRED',253,'PROXY COPY AVAILABLE',252,'PROXY COPY EXPIRED',251,'BACKUP
PIECE UNAVAILABLE',250,'PROXY COPY UNAVAILABLE',249,'DATAFILE COPY
AVAILABLE',248,'DATAFILE COPY EXPIRED',247,'DATAFILE COPY

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
UNAVAILABLE',246,'ARCHIVED LOG AVAILABLE',245,'ARCHIVED LOG
EXPIRED',244,'ARCHIVED LOG UNAVAILABLE',243,'BACKUP SET KEEP
OPTIONS',242,'BACKUP SET KEEP UNTIL',241,'PROXY COPY KEEP OPTIONS',240,'PROXY
COPY KEEP UNTIL',239,'DATAFILE COPY KEEP OPTIONS',238,'DATAFILE COPY KEEP
UNTIL',237,'DATAFILE RENAME ON RESTORE',236,'TEMPFILE RENAME', 235,'PREVIOUS
PIECE BACKUP SET', 234,'PLUGGED READONLY RENAME', 233,'BACKUP RECORD CLEANUP',
'UNKNOWN'),dlobp,dlosm,dltsd,decode(dlbss, 0, to_number(NULL), dlbss),
decode(dlbss, 0, to_number(NULL), dlbsc) from x$kccdl


VIEW_NAME
------------------------------
```

VIEW_DEFINITION

GV$DETACHED_SESSION
SELECT indx, inst_id, ksupgsnm, ksupgsid, ksupgser, ksupgspd FROM x$ksupgs

GV$DIAG_INFO
SELECT inst_id, name, value FROM x$diag_info

GV$DISPATCHER
select inst_id,kmmdinam,kmmdiadd,kmmdipro,kmmdista,
decode(kmmdiacc,0,'NO','YES'),kmmdinmg,kmmdinmb,kmmdibrk,

VIEW_NAME

VIEW_DEFINITION
----
kmmdinvo,kmmditnc,kmmdiidl,kmmdibsy,kmmdiler,kmmdidci from x$kmmdi where
kmmdiflg != 0

GV$DISPATCHER_CONFIG
select inst_id, indx, kmmdpnet, kmmdpopt, kmmdpcon, kmmdpses,
decode(bitand(kmmdpflg, 3), 0, 'OFF', 1, 'IN', 2, 'OUT', 'BOTH'), kmmdptck,
kmmdptin, kmmdptou, decode(bitand(kmmdpflg, 12), 0, 'OFF', 4, 'IN', 8, 'OUT',
'BOTH'), kmmdplsn,kmmdpsnm from x$kmmdp

VIEW_NAME

VIEW_DEFINITION

GV$DISPATCHER_RATE
select inst_id,kmmdinam,kmmdipro,kmmdicrle,kmmdicre,kmmdicepl,kmmdicrm,
kmmdicrus,kmmdicrys,kmmdicyus,kmmdicruc,kmmdicryc,kmmdicyuc,kmmdicru,
kmmdicry,kmmdicyu,kmmdicic,kmmdicoc,kmmdicrr,kmmdimrle,kmmdimre,kmmdimepl,
kmmdimrm,kmmdimrus,kmmdimrys,kmmdimyus,kmmdimruc,kmmdimryc,kmmdimyuc,
kmmdimru,kmmdimry,kmmdimyu,kmmdimic,kmmdimoc,kmmdimrr,kmmdiarle,kmmdiare,
kmmdiaepl,kmmdiarm,kmmdiarus,kmmdiarys,kmmdiayus,kmmdiaruc,kmmdiaryc,
kmmdiayuc,kmmdiaru,kmmdiary,kmmdiayu,kmmdiaic,kmmdiaoc,kmmdiarr,
kmmdinrle,kmmdinrm,kmmdinrus,kmmdinruc,kmmdinru,kmmdinic,kmmdinoc,kmmdinrr,

VIEW_NAME

VIEW_DEFINITION
----
kmmdisrle,kmmdisrm,kmmdisrus,kmmdisruc,kmmdisru,kmmdisic,kmmdisoc,kmmdisrr from
x$kmmdi where kmmdiflg!=0

GV$DLM_ALL_LOCKS
select USERENV('Instance'), HANDLE, GRANT_LEVEL, REQUEST_LEVEL,
RESOURCE_NAME1, RESOURCE_NAME2, PID, TRANSACTION_ID0, TRANSACTION_ID1,
GROUP_ID, OPEN_OPT_DEADLOCK, OPEN_OPT_PERSISTENT,
OPEN_OPT_PROCESS_OWNED, OPEN_OPT_NO_XID, CONVERT_OPT_GETVALUE,

CONVERT_OPT_PUTVALUE, CONVERT_OPT_NOVALUE, CONVERT_OPT_DUBVALUE,

VIEW_NAME

VIEW_DEFINITION

CONVERT_OPT_NOQUEUE, CONVERT_OPT_EXPRESS, CONVERT_OPT_NODEADLOCKWAIT,
CONVERT_OPT_NODEADLOCKBLOCK, WHICH_QUEUE, STATE, AST_EVENT0,
OWNER_NODE, BLOCKED, BLOCKER from V$GES_ENQUEUE

GV$DLM_CONVERT_LOCAL
select inst_id, kjicvtnam, kjicvtalt, kjicvtalc from x$kjicvt

GV$DLM_CONVERT_REMOTE
select inst_id, kjicvtnam, kjicvtart, kjicvtarc from x$kjicvt

VIEW_NAME

VIEW_DEFINITION

GV$DLM_LATCH
select USERENV('Instance'), addr, latch#, level#, name, gets, misses,
sleeps,immediate_gets, immediate_misses, waiters_woken, waits_holding_latch,
spin_gets, sleep1, sleep2, sleep3, sleep4, sleep5, sleep6, sleep7, sleep8,
sleep9, sleep10, sleep11, wait_time from V$LATCH where NAME like 'ges %' or
NAME like 'gcs %'

GV$DLM_LOCKS

VIEW_NAME

VIEW_DEFINITION
---
select USERENV('Instance'), HANDLE, GRANT_LEVEL, REQUEST_LEVEL,
RESOURCE_NAME1, RESOURCE_NAME2, PID, TRANSACTION_ID0, TRANSACTION_ID1,
GROUP_ID, OPEN_OPT_DEADLOCK, OPEN_OPT_PERSISTENT,
OPEN_OPT_PROCESS_OWNED, OPEN_OPT_NO_XID, CONVERT_OPT_GETVALUE,
CONVERT_OPT_PUTVALUE, CONVERT_OPT_NOVALUE, CONVERT_OPT_DUBVALUE,
CONVERT_OPT_NOQUEUE, CONVERT_OPT_EXPRESS, CONVERT_OPT_NODEADLOCKWAIT,
CONVERT_OPT_NODEADLOCKBLOCK, WHICH_QUEUE, STATE, AST_EVENT0,
OWNER_NODE, BLOCKED, BLOCKER from V$GES_BLOCKING_ENQUEUE

VIEW_NAME

VIEW_DEFINITION

GV$DLM_MISC
select inst_id, indx, kjisftdesc, kjisftval from x$kjisft

GV$DLM_RESS

select inst_id, kjirftrp, kjirftrn, kjirftcq, kjirftgq, kjirftpr, kjirftmn,
kjirftncl, kjirftvs, kjirftvb from x$kjirft union all select inst_id, kjbrresp,
kjbrname, decode(kjbrcvtq, '00', 0, 1), decode(kjbrgrantq, '00', 0, 1), 1,
kjbrmaster, kjbrncvl, 'KJUSERVS_NOVALUE', '0x0' from x$kjbr

VIEW_NAME

VIEW_DEFINITION

GV$DLM_TRAFFIC_CONTROLLER
select inst_id, kjitrftlid, kjitrftrid, kjitrftrrd, kjitrftinc,
kjitrftta, kjitrfttl, kjitrfttr, decode(kjitrfttw,0, 'NO ',
'YES '), kjitrftss, kjitrftsr, kjitrftsql,
kjitrftsqm, kjitrftsqt, kjitrftqtb, kjitrftqtw, kjitrftst,
kjitrftpxy from x$kjitrft

GV$DNFS_CHANNELS
select inst_id, pnum, svrname, path, ch_id, svr_id, sends, recvs, pings

VIEW_NAME

VIEW_DEFINITION
----
from x$dnfs_channels

GV$DNFS_FILES
select inst_id, filename, filesize, pnum, svr_id from x$dnfs_files

GV$DNFS_SERVERS
select inst_id, id, svrname, dirname, mntport, nfsport, wtmax, rtmax
from x$dnfs_servers

VIEW_NAME

VIEW_DEFINITION

GV$DNFS_STATS
select inst_id, pnum, nfs_null, nfs_getattr, nfs_setattr, nfs_lookup,
nfs_access, nfs_readlink, nfs_read, nfs_write, nfs_create, nfs_mkdir,
nfs_symlink, nfs_mknod, nfs_remove, nfs_rmdir, nfs_rename, nfs_link,
nfs_readdir, nfs_readdirplus, nfs_fsstat, nfs_fsinfo, nfs_pathconf,
nfs_commit, nfs_mount from x$dnfs_stats

GV$DYNAMIC_REMASTER_STATS
select inst_id, drms, avg_drm_time, objects_per_drm, quiesce_t, frz_t,

VIEW_NAME

VIEW_DEFINITION

```
cleanup_t, replay_t, fixwrite_t, sync_t, res_cleaned, replay_s,  replay_r,
my_objects FROM x$kjdrmafnstats

GV$ENABLEDPRIVS
select inst_id,-kzsprprv from x$kzspr

GV$ENCRYPTED_TABLESPACES
select INST_ID, TS#,                    decode(ALG, 0, 'NONE',
1, '3DES168',                        2, 'AES128',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
3, 'AES192',                         4, 'AES256'),
decode(ENCTS, 0, 'NO', 'YES')              from X$KCBTEK where ENCTS <> 0
and            DROPPEDTS = 0

GV$ENCRYPTION_WALLET
SELECT INST_ID, WRL_TYPE, WRL_PARAMETER, decode(BITAND(STATUS,7), 1,
'UNDEFINED', 2, 'CLOSED', 4, 'OPEN') FROM X$KZEKMENCWAL

GV$ENQUEUE_LOCK

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select s.inst_id,l.addr,l.ksqlkadr,s.ksusenum,r.ksqrsidt,
r.ksqrsid1,r.ksqrsid2, l.ksqlkmod, l.ksqlkreq,l.ksqlkctim,l.ksqlklblk from
x$ksqeq l,x$ksuse s,x$ksqrs r where l.ksqlkses=s.addr and
bitand(l.kssobflg,1)!=0 and (l.ksqlkmod!=0 or l.ksqlkreq!=0) and
l.ksqlkres=r.addr

GV$ENQUEUE_STAT
select inst_id, ksqsttyp, sum(ksqstreq), sum(ksqstwat), sum(ksqstsgt),
sum(ksqstfgt), sum(ksqstwtm)    from X$KSQST group by inst_id, ksqsttyp having

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
sum(ksqstreq) > 0

GV$ENQUEUE_STATISTICS
select st.inst_id, eqt.name, st.ksqsttyp, st.ksqstrsn,          st.ksqstreq,
st.ksqstwat, st.ksqstsgt, st.ksqstfgt, st.ksqstwtm,          st.ksqstexpl,
st.ksqstevidx    from X$KSQST st, X$KSQEQTYP eqt    where (st.inst_id =
eqt.inst_id) and (st.ksqsttyp = eqt.resname) and          (st.indx > 0)

GV$EVENTMETRIC
```

VIEW_NAME

VIEW_DEFINITION

SELECT inst_id, begtime, endtime, intsize_csec, wait#, wait_id, nsess_wait, time_waited, wait_count FROM x$kewmevmv WHERE flag1 = 1 AND GROUPID = 0

GV$EVENT_HISTOGRAM
select d.inst_id, d.indx, d.kslednam, s.kslsesmaxdur, s.kslsesval from x$kslseshist s, x$ksled d where s.kslsesenum = d.indx

GV$EVENT_NAME

VIEW_NAME

VIEW_DEFINITION
---
select inst_id, indx, ksledhash, kslednam, ksledp1, ksledp2, ksledp3, ksledclassid, ksledclass#, ksledclass from x$ksled

GV$EXECUTION
select inst_id, pid, val0, func, decode(id,1,'call',2,'return',3,'longjmp'), nvals, val2, val3, seqh, seql from x$kstex where op=10

GV$FAST_START_SERVERS
SELECT inst_id, state, wdone, pid, xid from x$ktprxrs

VIEW_NAME

VIEW_DEFINITION

GV$FAST_START_TRANSACTIONS
SELECT inst_id, usn, slt, seq, state, wkd, twk, pid, etime, parentusn, parentslt, parentseq, xid, pxid, svrs from x$ktprxrt UNION ALL SELECT inst_id, usn, slt, seq, state, twk-wkl, twk, NULL, etime, NULL, NULL, NULL, xid, NULL, svrs from x$kturhist

GV$FILEMETRIC
SELECT inst_id, begtime, endtime, intsize_csec, fileid, creationtime,

VIEW_NAME

VIEW_DEFINITION
---
avrdtime, avwrtime, phyread, phywrite, phybkrd, phybkwr FROM x$kewmflmv WHERE flag1 = 1

GV$FILEMETRIC_HISTORY
SELECT inst_id, begtime, endtime, intsize_csec, fileid, creationtime, avrdtime, avwrtime, phyread, phywrite, phybkrd, phybkwr FROM

x$kewmflmv

GV$FILESPACE_USAGE

VIEW_NAME

VIEW_DEFINITION

SELECT inst_id, KTTEFINFOTSN, KTTEFINFOFNO, KTTEFINFOUSP,
KTTEFINFOSIZE, KTTEFINFOMSIZE, KTTEFINFOSCNB, KTTEFINFOSCNW,
KTTEFINFOFLAG FROM X$KTTEFINFO

GV$FILESTAT
select k.inst_id, k.kcfiofno,k.kcfiopyr,k.kcfiopyw,k.kcfiopbr,k.kcfiopbw,
k.kcfiosbr,k.kcfioprt,k.kcfiopwt,k.kcfiosbt,k.kcfioavg,k.kcfiolst,k.kcfiomin,
k.kcfiormx,k.kcfiowmx from x$kcfio k,x$kccfe f where f.fedup <> 0 and
f.fenum=k.kcfiofno

VIEW_NAME

VIEW_DEFINITION

GV$FILE_CACHE_TRANSFER
select x.inst_id, kcfiofno, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0 from x$kcfio x, x$kccfe fe where x.kcfiofno =
fe.fenum

GV$FILE_HISTOGRAM
select k.inst_id, k.kcfiofno,k.kcfiomaxdur,k.kcfioval from x$kcfiohist k,x$kccfe
f where f.fedup <> 0 and f.fenum=k.kcfiofno

VIEW_NAME

VIEW_DEFINITION

GV$FILE_PING
select x.inst_id, kcfiofno, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 from x$kcfio x, x$kccfe fe
where x.kcfiofno = fe.fenum

GV$FIXED_TABLE
select inst_id,kqftanam, kqftaobj, 'TABLE', indx from x$kqfta union all select
inst_id,kqfvinam, kqfviobj, 'VIEW', 65537 from x$kqfvi union all select

VIEW_NAME

VIEW_DEFINITION
---
inst_id,kqfdtnam, kqfdtobj, 'TABLE', 65537 from x$kqfdt

```
GV$FIXED_VIEW_DEFINITION
select i.inst_id,kqfvinam,kqftpsel from x$kqfvi i, x$kqfvt t where i.indx =
t.indx

GV$FLASHBACK_DATABASE_LOG
select inst_id,         to_number(fblogscn),         to_date(fblogtim,
'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),         fblogretn,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
totsize, to_number(fblogesiz)         from x$krfblog,         ( select
sum(flebsz * flenblks) totsize           from x$kccfle where fledup != 0 )

GV$FLASHBACK_DATABASE_LOGFILE
select fn.inst_id, fn.fnnam,              fle.flelno, fle.flethr,
fle.fleseq,              fle.flenblks * fle.flebsz,
to_number(fle.flelscn),              to_date(fle.fleltim, 'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian')         from x$kccfn fn, x$kccfle fle
where (fn.fntyp = 24)          and (fn.fnnum = fle.flefnh)          and

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
(fle.fledup != 0)

GV$FLASHBACK_DATABASE_STAT
select inst_id,  to_date(btime, 'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),
to_date(etime, 'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),  fbw*512,
dbw*512, redow*512, to_number(fbsz)  from x$krfgstat

GV$FOREIGN_ARCHIVED_LOG
select

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
inst_id,rlrid,rlstm,rlnam,rldst,rlthp,rlseq,to_number(rlrls),to_date(rlrlc,'MM/D
D/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(rlxlc),to_number(rllos),to_date(
rllot,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(rlnxs),to_date(rlnxt,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),rlbct,rlbsz,decode(bitand(rlflg,
16+32+64+128+256),    16, 'ARCH',    32, 'FGRD',    64, 'RMAN',    128,'SRMN',
256,'LGWR',        'UNKNOWN'),decode(bitand(rlflg, 4),    4,  'RFS',
decode(bitand(rlflg, 16+32+64+128+256),    16, 'ARCH',    32, 'FGRD',    64,

VIEW_NAME
------------------------------
```

```
VIEW_DEFINITION
--------------------------------------------------------------------------------
'RMAN',    128,'SRMN',    256,'LGWR',        'UNKNOWN')),decode(bitand(rlflg,
2),0,'NO','YES'),decode(bitand(rlflg, 1024),0,'NO','YES'),decode(bitand(rlflg,
1),0,'NO','YES'),decode(bitand(rlflg, 1+2048+4096),    0,    'A',    1,    'D',
2048,'X',    4096,'U',        '?'),to_date(rltsm,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(bitand(rlflg,8192),0,'NO','YES'),de
code(bitand(rlflg,16384),0,'NO','YES'),
decode(bitand(rlflg,32768),0,'NO','YES'),
rltoa,decode(bitand(rlfl2,64),0,'NO','YES'),
decode(bitand(rlfl2,128),0,'NO','YES'), decode(bitand(rlflg,512),0,'NO','YES'),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(bitand(rlfl2,256+512+1024),      256, 'TERMINAL',      512,
'ACTIVATION',      1024, 'RESETLOGS',
decode(bitand(rlflg,32768),0,'','SWITCHOVER')), rldbi from x$kccrl

GV$FS_FAILOVER_HISTOGRAM
select INST_ID,REDO_LATENCY,FREQUENCY,LAST_TIME from x$rfahist

GV$FS_FAILOVER_STATS
select INST_ID, FTIME, REASON from x$rfafo

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

GV$GCSHVMASTER_INFO
select inst_id,  KJDRPCMHVID, KJDRPCMHVCMAS, KJDRPCMHVPMAS, KJDRPCMHVRMCNT  from
x$kjdrpcmhv

GV$GCSPFMASTER_INFO
select inst_id,  KJDRPCMPFID, KJDRPCMPOID, KJDRPCMPTYPE, KJDRPCMPFCMAS,
KJDRPCMPFPMAS, KJDRPCMPFRMCNT  from x$kjdrpcmpf


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$GC_ELEMENT
select inst_id, le_addr, indx, le_class,          name, le_mode, le_blks,
le_rls, le_acq,          le_write, le_recovery, le_local, le_flags
from x$le

GV$GC_ELEMENTS_WITH_COLLISIONS
select USERENV('Instance'), lock_element_addr  from v$bh        where
(forced_writes + forced_reads) > 10           group by lock_element_addr
```

```
having count(*) >= 2

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

GV$GES_BLOCKING_ENQUEUE
select USERENV('Instance'), HANDLE, GRANT_LEVEL, REQUEST_LEVEL,
RESOURCE_NAME1, RESOURCE_NAME2, PID, TRANSACTION_ID0,         TRANSACTION_ID1,
GROUP_ID, OPEN_OPT_DEADLOCK, OPEN_OPT_PERSISTENT,
OPEN_OPT_PROCESS_OWNED, OPEN_OPT_NO_XID, CONVERT_OPT_GETVALUE,
CONVERT_OPT_PUTVALUE, CONVERT_OPT_NOVALUE,         CONVERT_OPT_DUBVALUE,
CONVERT_OPT_NOQUEUE,         CONVERT_OPT_EXPRESS, CONVERT_OPT_NODEADLOCKWAIT,
CONVERT_OPT_NODEADLOCKBLOCK, WHICH_QUEUE, STATE,         AST_EVENT0,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
OWNER_NODE, BLOCKED, BLOCKER         from V$GES_ENQUEUE         where
(REQUEST_LEVEL != 'KJUSERNL') and         (BLOCKED = 1 or BLOCKER = 1)

GV$GES_ENQUEUE
select inst_id, kjilkftlkp, kjilkftgl, kjilkftrl, kjilkftrn1, kjilkftrn2,
kjilkftpid, kjilkftxid0, kjilkftxid1, kjilkftgid, kjilkftoodd, kjilkftoopt,
kjilkftoopo, kjilkftoonxid, kjilkftcogv, kjilkftcopv, kjilkftconv, kjilkftcodv,
kjilkftconq, kjilkftcoep, kjilkftconddw, kjilkftconddb, kjilkftwq, kjilkftls,
kjilkftaste0, kjilkfton, kjilkftblked, kjilkftblker from x$kjilkft union all

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select inst_id, kjbllockp, kjblgrant, kjblrequest, kjblname, kjblname2, 0, 0, 0,
0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, kjblqueue, kjbllockst, 0, kjblowner,
kjblblocked, kjblblocker from x$kjbl

GV$GLOBALCONTEXT
select namespace, attribute, value,         username,clientidentifier
from x$globalcontext         where upper(namespace) not like 'SYS_%'

GV$GLOBAL_BLOCKED_LOCKS

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select USERENV('instance'), addr, kaddr, sid, type, id1, id2,
lmode,request,ctime from v$lock l where exists (select * from v$dlm_locks d
where substr(d.resource_name2,1,instr(d.resource_name2, ',',1,1)-1) = id1 and
substr(d.resource_name2,instr(d.resource_name2,',',1,1)+1,
```

```
instr(d.resource_name2 , ',',1,2)-instr(d.resource_name2, ',',1,1)-1) = id2 and
substr(d.resource_name2,instr(d.resource_name2, ',',-1,1)+1,2) = type)

GV$GLOBAL_TRANSACTION
select inst_id,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
K2GTIFMT, K2GTITID_EXT, K2GTIBID, K2GTECNT, K2GTERCT, K2GTDPCT,
decode (K2GTDFLG, 0, 'ACTIVE', 1, 'COLLECTING', 2, 'FINALIZED',
4, 'FAILED', 8, 'RECOVERING', 16, 'UNASSOCIATED',
32, 'FORGOTTEN', 64, 'READY FOR RECOVERY',
128, 'NO-READONLY FAILED', 256, 'SIBLING INFO WRITTEN',
512,      '[ORACLE COORDINATED]ACTIVE',
512+1,    '[ORACLE COORDINATED]COLLECTING',
512+2,    '[ORACLE COORDINATED]FINALIZED',
512+4,    '[ORACLE COORDINATED]FAILED',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
512+8,    '[ORACLE COORDINATED]RECOVERING',
512+16,   '[ORACLE COORDINATED]UNASSOCIATED',
512+32,   '[ORACLE COORDINATED]FORGOTTEN',
512+64,   '[ORACLE COORDINATED]READY FOR RECOVERY',
512+128,  '[ORACLE COORDINATED]NO-READONLY FAILED',
1024,     '[MULTINODE]ACTIVE',
1024+1,   '[MULTINODE]COLLECTING',
1024+2,   '[MULTINODE]FINALIZED',
1024+4,   '[MULTINODE]FAILED',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
1024+8,   '[MULTINODE]RECOVERING',
1024+16,  '[MULTINODE]UNASSOCIATED',
1024+32,  '[MULTINODE]FORGOTTEN',
1024+64,  '[MULTINODE]READY FOR RECOVERY',
1024+128, '[MULTINODE]NO-READONLY FAILED',
1024+256, '[MULTINODE]SIBLING INFO WRITTEN',
'COMBINATION'), K2GTDFLG,                              decode (K2GTETYP, 0,
'FREE', 1, 'LOOSELY COUPLED', 2, 'TIGHTLY COUPLED')      from X$K2GTE2


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
```

GV$HM_CHECK
```
select inst_id,      id,      name,      clsid,      decode(clsid, 1,
'GENERIC',                2, 'PERSISTENT_DATA',
'UNKNOWN'),      flags,      decode(bitand(flags,1), 0, 'N', 'Y'),
decode(bitand(flags,2), 0, 'N', 'Y'),      description from  x$dbkh_check
where bitand(flags,64) = 0
```

GV$HM_CHECK_PARAM
```
select inst_id,      id,      name,      check_id,      decode(type, 0,
```

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
'DBKH_PARAM_UB4',                1, 'DBKH_PARAM_UB8',
2, 'DBKH_PARAM_TEXT',                3, 'DBKH_PARAM_DATE',
4, 'DBKH_PARAM_UB4_LIST',                5, 'DBKH_PARAM_UB8_LIST',
6, 'DBKH_PARAM_TEXT_LIST',                7, 'DBKH_PARAM_DATE_LIST',
'UNKNOWN'),      default_value,      flags,      description from
x$dbkh_check_param
```

GV$HM_FINDING
```
select inst_id,      id,      run_id,      name,      pid,      cid_count,
```

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
clsname,      cast(ctime as timestamp),      cast(mtime as timestamp),
decode(priority, 0, 'CRITICAL',                1, 'HIGH',
2, 'LOW',                'UNKNOWN'),      decode(status, 0, 'OPEN',
1, 'CLOSED',                2, 'UNDER-REPAIR',
'UNKNOWN'),      decode(type, 0, 'INFORMATIONAL',                1,
'FAILURE',                'UNKNOWN'),      fdg_msg,      damage_msg
from x$dbkfdg
```

GV$HM_INFO

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select inst_id,      id,      decode(type, 0, 'RUN',                1,
'RUN-RESUME',                2, 'FINDING',                3,
'RECOMMENDATION',                'UNKNOWN'),      name,      value
from x$dbkinfo
```

GV$HM_RECOMMENDATION
```
select inst_id,      id,      fid,      runid,      name,      decode(type,
0, 'MANUAL',                1, 'REAPIR',                'UNKNOWN'),
rank,      cast(ctime as timestamp),      cast(etime as timestamp),
```

VIEW_NAME

VIEW_DEFINITION
---
decode(status, 0, 'NOT RUN', 1, 'RUNNING',
2, 'SUCCESS', 3, 'FAILED',
'UNKNOWN'), reco_msg, script from x$dbkreco

GV$HM_RUN
select inst_id, id, name, cname, decode(runmode, 0,
'MANUAL', 1, 'AUTO', 2,
'REACTIVE', 'UNKNOWN'), timeout, cast(stime
as timestamp), cast(rtime as timestamp), cast(etime as timestamp),

VIEW_NAME

VIEW_DEFINITION
---
cast(mtime as timestamp), decode(status, 0, 'INITITAL',
1, 'EXECUTING', 2, 'INTERRUPTED', 3,
'TIMEDOUT', 4, 'CANCELLED', 5,
'COMPLETED', 6, 'ERROR', 'UNKNOWN'),
incident, num_incidents, error_number, problem_id from
x$dbkrun

GV$HS_AGENT
select unique INST_ID, AGENT_ID, MACHINE, PROCESS, PROGRAM, OSUSER,

VIEW_NAME

VIEW_DEFINITION

AGT_STARTTIME, AGENT_TYPE, decode(AGENT_TYPE, 1, to_number(NULL),
FDS_CLASS_ID), decode(AGENT_TYPE, 1, to_number(NULL), FDS_INST_ID)
from X$HS_SESSION

GV$HS_PARAMETER
select A.INST_ID, HS_SESSION_ID, PARAMETER, VALUE, SOURCE, ENV from
X$HS_SESSION A, X$HOFP B WHERE A.FDS_INST_ID = B.FDS_INST_ID

GV$HS_SESSION

VIEW_NAME

VIEW_DEFINITION
---
select INST_ID, HS_SESSION_ID, AGENT_ID, SID, decode(
AGENT_TYPE, 1, NULL, DB_LINK), decode(AGENT_TYPE, 1, to_number(NULL),
DB_LINK_OWNER), SES_STARTTIME from X$HS_SESSION

GV$HVMASTER_INFO
select inst_id, KJDRHVID, KJDRHVCMAS, KJDRHVPMAS, KJDRHVRMCNT from x$kjdrhv

```
GV$INCMETER_CONFIG
select inst_id,      tilt,      cfactor,      wfactor,      wtfactor,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
mtime from x$dbkincmetcfg

GV$INCMETER_INFO
select inst_id,      incident_id,      ctime,      decode(is_disabled,  0,
'N',                      1, 'Y'),      decode(is_active,    0, 'N',
1, 'Y'),      decode(is_pers_impt, 0, 'TRANSIENT',
1, 'PERSISTENT',                      'UNKNOWN      '),
impact1,      impact2,      impact3,      impact4 from x$dbkincmetinfo


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$INCMETER_SUMMARY
select inst_id,      decode(severity_idx, 0, 'NORMAL',
1, 'WARNING',                      2, 'CRITICAL',
'UNKNOWN      '),      critical_incidents,      warning_incidents,
last_hour_incidents,      ctime,      otictime,      opictime,
lictime from x$dbkincmetsummary

GV$INDEXED_FIXED_COLUMN
select c.inst_id,kqftanam, kqfcoidx, kqfconam, kqfcoipo from x$kqfco c, x$kqfta

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
t where t.indx = c.kqfcotab and kqfcoidx != 0

GV$INSTANCE
select
ks.inst_id,ksuxsins,ksuxssid,ksuxshst,ksuxsver,ksuxstim,decode(ksuxssts,0,'START
ED',1,'MOUNTED',2,'OPEN',3,'OPEN
MIGRATE','UNKNOWN'),decode(ksuxsshr,0,'NO',1,'YES',2,NULL),ksuxsthr,decode(ksuxs
arc,0,'STOPPED',1,'STARTED','FAILED'),decode(ksuxslsw,0,NULL,2,'ARCHIVE
LOG',3,'CLEAR LOG',4,'CHECKPOINT',      5,'REDO

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GENERATION'),decode(ksuxsdba,0,'ALLOWED','RESTRICTED'),decode(ksuxsshp,0,'NO','Y
ES'),decode(kvitval,0,'ACTIVE',2147483647,'SUSPENDED','INSTANCE
```

```
RECOVERY'),decode(ksuxsrol,1,'PRIMARY_INSTANCE',2,'SECONDARY_INSTANCE','UNKNOWN'
), decode(qui_state,0,'NORMAL',1,'QUIESCING',2,'QUIESCED','UNKNOWN'),
decode(bitand(ksuxsdst, 1), 0, 'NO', 1, 'YES', 'NO') from x$ksuxsinst ks, x$kvit
kv, x$quiesce qu where kvittag = 'kcbwst'

GV$INSTANCE_CACHE_TRANSFER
select inst_id, instance,  decode(class,1,'data block',2,'sort block',3,'save

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
undo block',  4,'segment header',5,'save undo header',6,'free list',7,'extent
map',  8,'1st level bmb',9,'2nd level bmb',10,'3rd level bmb',  11,'bitmap
block',12,'bitmap index block',13,'file header block',14,'unused',  15,'undo
header',16,'undo block'),  lost, lost_time,  cr_2hop + cr_3hop, cr_2hop_time +
cr_3hop_time,  cr_2hop, cr_2hop_time, cr_3hop, cr_3hop_time,  cr_busy,
cr_busy_time, cr_congested, cr_congested_time,  current_2hop + current_3hop,
current_2hop_time + current_3hop_time,  current_2hop, current_2hop_time,
current_3hop, current_3hop_time,  current_busy, current_busy_time,
current_congested, current_congested_time   from x$instance_cache_transfer

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

GV$INSTANCE_LOG_GROUP
select  USERENV('Instance'), THREAD# , STATUS , ENABLED , GROUPS , INSTANCE ,
OPEN_TIME , CURRENT_GROUP# , SEQUENCE# , CHECKPOINT_CHANGE# , CHECKPOINT_TIME ,
ENABLE_CHANGE# , ENABLE_TIME , DISABLE_CHANGE# , DISABLE_TIME from V$THREAD

GV$INSTANCE_RECOVERY
select        T.INST_ID,        to_number(decode(CUR_EST_RCV_READS, -1,
NULL, CUR_EST_RCV_READS)),        to_number(decode(ACTUAL_REDO_BLKS, -1, NULL,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
ACTUAL_REDO_BLKS)),        to_number(decode(MIN_LAG, 0, NULL, MIN_LAG)),
to_number(decode(LOGFILESZ, 0, NULL, LOGFILESZ)),
to_number(decode(CT_LAG, 0, NULL, CT_LAG)),        to_number(decode(CI_LAG, 0,
NULL, CI_LAG)),        to_number(decode(ACTUAL_REDO_BLKS, 0, NULL, NULL)),
INUSE_NONRAC_MTTR_SEC+INUSE_RAC_MTTR_SEC,
to_number(decode(CUR_EST_IR_SEC, -1, CUR_EST_NONRAC_MTTR_SEC,
CUR_EST_IR_SEC)),        (select ksusgstv - (select ksusgstv from X$KSUSGSTA
where        ksusdnam='physical writes non checkpoint' and inst_id=t.inst_id)
from X$KSUSGSTA where ksusdnam = 'physical writes' and

VIEW_NAME
------------------------------
```

```
VIEW_DEFINITION
--------------------------------------------------------------------------------
inst_id=t.inst_id),         (select logfile_size from x$kctlax where indx=0 and
inst_id=t.inst_id),         to_number(decode(CUR_EST_CA_SEC, -1, NULL,
CUR_EST_CA_SEC)),         MTTR_WRITES, LOGFILE_SIZE_WRITES,
CKPT_SETTING_WRITES, OTHER_WRITES,         AUTO_WRITES, FULL_WRITES
from X$TARGETRBA T, X$ESTIMATED_MTTR E, X$KCTICW W where
T.INST_ID=E.INST_ID AND T.INST_ID=W.INST_ID

GV$IOFUNCMETRIC
SELECT inst_id, begtime, endtime, intsize_csec,              f.function_id,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
fn.function_name,              small_read_mbps, small_write_mbps,
large_read_mbps, large_write_mbps,              small_read_iops,
small_write_iops,              large_read_iops, large_write_iops,
decode(num_waits, 0, 0, wait_time / num_waits)         FROM   x$kewmiofmv f,
(select function_id, function_name from v$iostat_function
group by function_id, function_name) fn         WHERE flag1 = 1          AND
f.function_id = fn.function_id

GV$IOFUNCMETRIC_HISTORY

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
SELECT inst_id, begtime, endtime, intsize_csec,              f.function_id,
fn.function_name,              small_read_mbps, small_write_mbps,
large_read_mbps, large_write_mbps,              small_read_iops,
small_write_iops,              large_read_iops, large_write_iops,
decode(num_waits, 0, 0, wait_time / num_waits)         FROM   x$kewmiofmv f,
(select function_id, function_name from v$iostat_function
group by function_id, function_name) fn         WHERE f.function_id =
fn.function_id


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$IOSTAT_CONSUMER_GROUP
SELECT A.inst_id, CONSUMER_GROUP_ID_KSFDSTCG,
sum(round((SBRDATA_KSFDSTCG + A_SBRDATA_KSFDSTCG) / 2048)),
sum(round((SBWDATA_KSFDSTCG + A_SBWDATA_KSFDSTCG) / 2048)),
sum(round((MBRDATA_KSFDSTCG + A_MBRDATA_KSFDSTCG) / 2048)),
sum(round((MBWDATA_KSFDSTCG + A_MBWDATA_KSFDSTCG) / 2048)),
sum(SBRREQS_KSFDSTCG + A_SBRREQS_KSFDSTCG),
sum(SBWREQS_KSFDSTCG + A_SBWREQS_KSFDSTCG),
```

```
sum(MBRREQS_KSFDSTCG + A_MBRREQS_KSFDSTCG),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
sum(MBWREQS_KSFDSTCG + A_MBWREQS_KSFDSTCG),
sum(WREQS_KSFDSTCG + A_WREQS_KSFDSTCG),
sum(WTIME_KSFDSTCG + A_WTIME_KSFDSTCG)
FROM X$KSFDSTCG A , x$kgskcft B
WHERE B.class_id_kgskcft = A.consumer_group_id_ksfdstcg
GROUP BY A.INST_ID, CONSUMER_GROUP_ID_KSFDSTCG

GV$IOSTAT_FILE
SELECT k.inst_id, k.FILENO_KSFDSTFILE, 2, 'Data File',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
round(k.SBRDATA_KSFDSTFILE / 2048),
round(k.SBWDATA_KSFDSTFILE / 2048),
round(k.MBRDATA_KSFDSTFILE / 2048),
round(k.MBWDATA_KSFDSTFILE / 2048),
k.SBRSREQS_KSFDSTFILE, k.SBWSREQS_KSFDSTFILE,
k.SSBRREQS_KSFDSTFILE, k.MBRSREQS_KSFDSTFILE,
k.MBWSREQS_KSFDSTFILE, k.SBRSERV_KSFDSTFILE,
k.SBWSERV_KSFDSTFILE, k.SSBRLATENCY_KSFDSTFILE,
k.MBRSERV_KSFDSTFILE, k.MBWSERV_KSFDSTFILE,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(bitand(k.FLAGS_KSFDSTFILE, 4), 0, 'ASYNC_OFF', 'ASYNC_ON'),
k.RETRIES_KSFDSTFILE
FROM X$KSFDSTFILE k, x$kccfe f where f.fedup <> 0
and f.fenum=k.FILENO_KSFDSTFILE and k.FILETYPE_KSFDSTFILE=2
union
SELECT k.inst_id, k.FILENO_KSFDSTFILE, 6, 'Temp File',
round(k.SBRDATA_KSFDSTFILE / 2048),
round(k.SBWDATA_KSFDSTFILE / 2048),
round(k.MBRDATA_KSFDSTFILE / 2048),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
round(k.MBWDATA_KSFDSTFILE / 2048),
k.SBRSREQS_KSFDSTFILE, k.SBWSREQS_KSFDSTFILE,
k.SSBRREQS_KSFDSTFILE, k.MBRSREQS_KSFDSTFILE,
k.MBWSREQS_KSFDSTFILE,
```

```
round(k.SBRSERV_KSFDSTFILE / 1000),
round(k.SBWSERV_KSFDSTFILE / 1000),
round(k.SSBRLATENCY_KSFDSTFILE / 1000),
round(k.MBRSERV_KSFDSTFILE / 1000),
round(k.MBWSERV_KSFDSTFILE / 1000),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(bitand(k.FLAGS_KSFDSTFILE, 4), 0, 'ASYNC_OFF', 'ASYNC_ON'),
k.RETRIES_KSFDSTFILE
FROM X$KSFDSTFILE k, x$kcctf f where f.tfdup <> 0
and f.tfnum=k.FILENO_KSFDSTFILE and k.FILETYPE_KSFDSTFILE=6
union
SELECT k.inst_id, k.FILENO_KSFDSTFILE, k.FILETYPE_KSFDSTFILE,
decode(k.FILETYPE_KSFDSTFILE,
1,'Control File', 2,'Data File',3,'Log File',4,'Archive Log',
6,'Temp File', 9,'Data File Backup',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
10,'Data File Incremental Backup',
11,'Archive Log Backup', 12,'Data File Copy', 17,'Flashback Log',
18,'Data Pump Dump File', 'Other'),
round(k.SBRDATA_KSFDSTFILE / 2048),
round(k.SBWDATA_KSFDSTFILE / 2048),
round(k.MBRDATA_KSFDSTFILE / 2048),
round(k.MBWDATA_KSFDSTFILE / 2048),
k.SBRSREQS_KSFDSTFILE, k.SBWSREQS_KSFDSTFILE,
k.SSBRREQS_KSFDSTFILE, k.MBRSREQS_KSFDSTFILE,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
k.MBWSREQS_KSFDSTFILE,
round(k.SBRSERV_KSFDSTFILE / 1000),
round(k.SBWSERV_KSFDSTFILE / 1000),
round(k.SSBRLATENCY_KSFDSTFILE / 1000),
round(k.MBRSERV_KSFDSTFILE / 1000),
round(k.MBWSERV_KSFDS

GV$IOSTAT_FUNCTION
SELECT inst_id, COMPONENT_ID_KSFDSTCMP,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
```

```
decode(COMPONENT_ID_KSFDSTCMP,                                    0,
'RMAN', 1, 'DBWR', 2, 'LGWR', 3, 'ARCH',                          4, 'XDB',
5, 'Streams AQ',                                  6, 'Data Pump',
7, 'Recovery', 8, 'Buffer Cache Reads',             9, 'Direct Reads', 10,
'Direct Writes',                          11, 'Others'),
sum(round((SBRDATA_KSFDSTCMP + A_SBRDATA_KSFDSTCMP) / 2048)),
sum(round((SBWDATA_KSFDSTCMP + A_SBWDATA_KSFDSTCMP) / 2048)),
sum(round((MBRDATA_KSFDSTCMP + A_MBRDATA_KSFDSTCMP) / 2048)),
sum(round((MBWDATA_KSFDSTCMP + A_MBWDATA_KSFDSTCMP) / 2048)),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
sum(SBRREQS_KSFDSTCMP + A_SBRREQS_KSFDSTCMP),
sum(SBWREQS_KSFDSTCMP + A_SBWREQS_KSFDSTCMP),
sum(MBRREQS_KSFDSTCMP + A_MBRREQS_KSFDSTCMP),
sum(MBWREQS_KSFDSTCMP + A_MBWREQS_KSFDSTCMP),
sum(WREQS_KSFDSTCMP + A_WREQS_KSFDSTCMP),
sum(WTIME_KSFDSTCMP + A_WTIME_KSFDSTCMP)                          FROM
X$KSFDSTCMP GROUP BY inst_id, COMPONENT_ID_KSFDSTCMP

GV$IOSTAT_NETWORK

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select  INST_ID,      KSRPCCLIENT,      KSRPCREADS,      KSRPCWRITES,
KSRPCRKB,      KSRPCWKB,      KSRPCRLATENCY,      KSRPCWLATENCY from
X$KSRPCIOS

GV$IO_CALIBRATION_STATUS
SELECT inst_id,
decode(status_kkkicr, 1, 'IN PROGRESS',
2, 'READY',                                    'NOT
AVAILABLE'),              endtime_kkkicr

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
FROM X$KKKICR

GV$IR_FAILURE
select inst_id,      id,      pid,      cid_count,      clsname,
cast(ctime as date),      cast(mtime as date),      fdg_msg,      damage_msg,
decode(priority, 0, 'CRITICAL',                   1, 'HIGH',
2, 'LOW',                   'UNKNOWN'),      decode(status, 0, 'OPEN',
1, 'CLOSED',                  2, 'UNDER-REPAIR',
'UNKNOWN') from x$dbkfdg    where type = 1 and clsnameid = 2 and bitand(FLAGS,1)
```

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
= 0

GV$IR_FAILURE_SET
select inst_id,      id,      failid,decode(bitand(flags,8), 0, 'NO', 'YES')
from x$dbkfset

GV$IR_MANUAL_CHECKLIST
select inst_id,      fid,      rank,decode(bitand(flags,65536), 0, 'NO',
'YES'),      reco_msg from x$dbkreco where type = 0

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

GV$IR_REPAIR
select inst_id,      id,      fid,      name,      rank,      cast(ctime as
date),      cast(etime as date),      cast('UNKNOWN' as varchar2(20)),
reco_msg,      script,      est_rtime,      act_rtime,      decode(status,
0, 'NOT RUN',                  1, 'RUNNING',                  2,
'SUCCESS',                  3, 'FAILED',                  'UNKNOWN')
from x$dbkreco where type = 1


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$JAVAPOOL
select inst_id, ksmchcom, sum(ksmchsiz) from x$ksmjch  group by inst_id,
ksmchcom order by ksmchcom

GV$JAVA_LIBRARY_CACHE_MEMORY
select inst_id,  decode(kgljsim_namespace, 0,'SQL AREA', 1,'TABLE/PROCEDURE',
2,'BODY', 3,'TRIGGER', 4,'INDEX', 5,'CLUSTER', 6,'OBJECT',        7,'PIPE',
13,'JAVA SOURCE', 14,'JAVA RESOURCE',        32,'JAVA DATA', '?'),
kgljsim_pincnt, kgljsim_pinmem, kgljsim_unpincnt, kgljsim_unpinmem  from

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
x$kgljmem where kgljsim_namespace<8 or kgljsim_namespace=13 or
kgljsim_namespace=14 or kgljsim_namespace=32  union  select inst_id,
'OTHER/SYSTEM', sum(kgljsim_pincnt) sum_pincnt,  sum(kgljsim_pinmem) sum_pinmem,
sum(kgljsim_unpincnt) sum_unpincnt,  sum(kgljsim_unpinmem) sum_unpinmem from
x$kgljmem where not  (kgljsim_namespace<8 or kgljsim_namespace=13 or
kgljsim_namespace=14  or kgljsim_namespace=32) group by inst_id
```

GV$JAVA_POOL_ADVICE
select inst_id, java_size, round(java_size / basejava_size, 4), kgljsim_size,

VIEW_NAME

VIEW_DEFINITION
---
kgljsim_objs, kgljsim_timesave, decode(kgljsim_basetimesave, 0,
to_number(null), round(kgljsim_timesave / kgljsim_basetimesave, 4)),
kgljsim_parsetime, decode(kgljsim_baseparsetime, 0, to_number(null),
round(kgljsim_parsetime / kgljsim_baseparsetime, 4)), kgljsim_hits from
x$kgljsim

GV$LATCH
select lt.inst_id,lt.kslltaddr,lt.kslltnum,lt.kslltlvl,lt.kslltnam,
lt.ksllthsh,lt.kslltwgt,lt.kslltwff,

VIEW_NAME

VIEW_DEFINITION
---
lt.kslltwsl,lt.kslltngt,lt.kslltnfa,lt.kslltwkc,
lt.kslltwth,lt.ksllthst0,lt.ksllthst1,lt.ksllthst2,
lt.ksllthst3,lt.ksllthst4,lt.ksllthst5,lt.ksllthst6,lt.ksllthst7,
lt.ksllthst8,lt.ksllthst9,lt.ksllthst10, lt.ksllthst11, lt.kslltwtt from
x$kslltr lt

GV$LATCHHOLDER
select inst_id,ksuprpid,ksuprsid,ksuprlat,ksuprlnm,ksulagts from x$ksuprlat

VIEW_NAME

VIEW_DEFINITION

GV$LATCHNAME
select inst_id,indx, kslldnam, kslldhsh from x$kslld

GV$LATCH_CHILDREN
select t.inst_id,t.kslltaddr,t.kslltnum,t.kslltcnm,t.kslltlvl,
t.kslltnam,t.ksllthsh,
t.kslltwgt,t.kslltwff,t.kslltwsl,t.kslltngt,t.kslltnfa,
t.kslltwkc,t.kslltwth,t.ksllthst0,t.ksllthst1,
t.ksllthst2,t.ksllthst3,t.ksllthst4,t.ksllthst5,

VIEW_NAME

VIEW_DEFINITION
---
t.ksllthst6,t.ksllthst7,t.ksllthst8, t.ksllthst9,t.ksllthst10,
t.ksllthst11, t.kslltwtt from x$kslltr_children t

GV$LATCH_MISSES
select t1.inst_id,t1.ksllasnam, t2.ksllwnam, t1.kslnowtf, t1.kslsleep,
t1.kslwscwsl, t1.kslwsclthg, t2.ksllwnam from x$ksllw t2, x$kslwsc t1
where t2.indx = t1.indx

GV$LATCH_PARENT

VIEW_NAME

VIEW_DEFINITION
---
select t.inst_id,t.kslltaddr,t.kslltnum,t.kslltlvl,t.kslltnam,t.kslthsh,
t.kslltwgt,t.kslltwff,t.kslltwsl,t.kslltngt,t.kslltnfa,
t.kslltwkc,t.kslltwth,t.kslthst0,t.kslthst1,
t.kslthst2,t.kslthst3,t.kslthst4,t.kslthst5,
t.kslthst6,t.kslthst7,t.kslthst8, t.kslthst9,t.kslthst10,
t.kslthst11, t.kslltwtt from x$kslltr_parent t

GV$LIBRARYCACHE
select inst_id, decode(indx,0,'SQL

VIEW_NAME

VIEW_DEFINITION

AREA',1,'TABLE/PROCEDURE',2,'BODY',3,'TRIGGER',
4,'INDEX',5,'CLUSTER',6,'OBJECT',7,'PIPE', 13,'JAVA SOURCE',14,'JAVA
RESOURCE',32,'JAVA DATA','?'), kglstget,kglstght,
decode(kglstget,0,1,kglstght/kglstget),kglstpin,kglstpht,
decode(kglstpin,0,1,kglstpht/kglstpin),kglstrld,kglstinv,
kglstlrq,kglstprq,kglstprl,kglstirq,kglstmiv from x$kglst where indx<8 or
indx=13 or indx=14 or indx=32

GV$LIBRARY_CACHE_MEMORY

VIEW_NAME

VIEW_DEFINITION
---
select inst_id, decode(kglsim_namespace, 0,'SQL AREA', 1,'TABLE/PROCEDURE',
2,'BODY', 3,'TRIGGER', 4,'INDEX', 5,'CLUSTER', 6,'OBJECT', 7,'PIPE',
13,'JAVA SOURCE', 14,'JAVA RESOURCE', 32,'JAVA DATA', '?'),
kglsim_pincnt, kglsim_pinmem, kglsim_unpincnt, kglsim_unpinmem from x$kglmem
where kglsim_namespace<8 or kglsim_namespace=13 or kglsim_namespace=14 or
kglsim_namespace=32 union select inst_id, 'OTHER/SYSTEM', sum(kglsim_pincnt)
sum_pincnt, sum(kglsim_pinmem) sum_pinmem, sum(kglsim_unpincnt) sum_unpincnt,
sum(kglsim_unpinmem) sum_unpinmem from x$kglmem where not (kglsim_namespace<8
or kglsim_namespace=13 or kglsim_namespace=14 or kglsim_namespace=32) group by

VIEW_NAME

VIEW_DEFINITION
---
inst_id

GV$LICENSE
select inst_id,ksullms,ksullws,ksullcs,ksullhs,ksullmu,cpu_count,
decode(cpu_core_count,0,to_number(null),cpu_core_count),
decode(cpu_socket_count,0,to_number(null),cpu_socket_count), cpu_count_hwm,
decode(cpu_core_count_hwm,0,to_number(null),cpu_core_count_hwm),
decode(cpu_socket_count_hwm,0,to_number(null),cpu_socket_count_hwm) from x$ksull

VIEW_NAME

VIEW_DEFINITION

GV$LOADISTAT
select inst_id,klcieon,klcietn,klciein,klcieisn,klciemno,klciemsg from x$klcie

GV$LOADPSTAT
select inst_id,klcpxon,klcpxtn,klcpxpn,klcpxrld from x$klpt

GV$LOBSTAT
select inst_id, LOBTSN, LOBRDBA, LOBOBJID, LOBCURRTIME, LOBEXPMQL, LOBSQLMQL,
LOBSPCANALTIME, LOBUNDORETTIME from x$lobstat

VIEW_NAME

VIEW_DEFINITION

GV$LOCK
select s.inst_id, l.laddr, l.kaddr, s.ksusenum, r.ksqrsidt, r.ksqrsid1,
r.ksqrsid2, l.lmode, l.request,l.ctime, decode(l.lmode, 0, 0,
l.block) from v$_lock l,x$ksuse s,x$ksqrs r where l.saddr=s.addr and
l.raddr=r.addr

GV$LOCKED_OBJECT
select x.inst_id,x.kxidusn, x.kxidslt, x.kxidsqn, l.ktadmtab, s.indx,

VIEW_NAME

VIEW_DEFINITION
---
s.ksuudlna, s.ksuseunm, s.ksusepid, l.ksqlkmod from x$ktcxb x, x$ktadm l,
x$ksuse s where x.ktcxbxba = l.kssobown and x.ktcxbses = s.addr

GV$LOCKS_WITH_COLLISIONS
select USERENV('Instance'), lock_element_addr from v$bh where
(forced_writes + forced_reads) > 10 group by lock_element_addr
having count(*) >= 2

GV$LOCK_ACTIVITY

VIEW_NAME

VIEW_DEFINITION
---
select 0, 'NULL', 'S', 'Lock buffers for read', 0 from dual

GV$LOCK_ELEMENT
select inst_id, le_addr, indx, le_class, name, le_mode, le_blks, le_rls,
le_acq, 0, le_flags from x$le

GV$LOCK_TYPE
select rest.inst_id, rest.resname, rest.name, rest.id1, rest.id2,
decode(bitand(eqt.flags, 1), 1, 'YES', 'NO'), rest.expl from X$KSIRESTYP

VIEW_NAME

VIEW_DEFINITION
---
rest, X$KSQEQTYP eqt where (rest.inst_id = eqt.inst_id) and (rest.indx =
eqt.indx) and (rest.indx > 0)

GV$LOG
select le.inst_id, le.lenum, le.lethr, le.leseq, le.lesiz*le.lebsz, ledup,
decode(bitand(le.leflg,1),0,'NO','YES'), decode(bitand(le.leflg,24), 8,
'CURRENT', 16,'CLEARING',
24,'CLEARING_CURRENT', decode(sign(leseq),0,'UNUSED',
decode(sign((to_number(rt.rtckp_scn)-to_number(le.lenxs))*

VIEW_NAME

VIEW_DEFINITION
---
bitand(rt.rtsta,2)),-1,'ACTIVE','INACTIVE'))), to_number(le.lelos),
to_date(le.lelot,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian') from x$kccle
le, x$kccrt rt where le.ledup!=0 and le.lethr=rt.rtnum and le.inst_id =
rt.inst_id

GV$LOGFILE
select inst_id,fnfno, decode(fnflg,0,'', decode(bitand(fnflg,1),1,'INVALID',
decode(bitand(fnflg,2),2,'STALE', decode(bitand(fnflg,4),4,'DELETED',
decode(bitand(fnflg,8+32),8,'',32,'',40,'','UNKNOWN'))))),

VIEW_NAME

VIEW_DEFINITION
---
decode(bitand(fnflg,8),0,'ONLINE','STANDBY'), fnnam, decode(bitand(fnflg,
32),0,'NO','YES') from x$kccfn where fnnam is not null and fntyp=3

GV$LOGHIST

```
select inst_id,lhthp,lhseq,to_number(lhlos), to_date(lhlot,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'), to_number(lhnxs) from x$kcclh

GV$LOGMNR_CALLBACK
select inst_id, session_id, function_id, description,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(type, 0, 'MISSING_LOGFILE', 1, 'ERROR',                          2,
'OBJECT_FILTER', 3, 'CHGVEC_READ',                          4,
'COMPLETE_LCR', 5, 'TXN_COMMIT',                          6,
'PROCESS_STATE', 'UNKNOWN')         from x$logmnr_callback

GV$LOGMNR_CONTENTS
select INST_ID, SCN, START_SCN, CSCN,         TIMESTAMP, START_TIMESTAMP,
COMMIT_TIMESTAMP,         XIDUSN, XIDSLT, XIDSQN, XID,         PXIDUSN, PXIDSLT,
PXIDSQN, PXID,         TX_NAME,         OPERATION, OPERATION_CODE, ROLLBACK,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
SEG_OWNER, SEG_NAME, TABLE_NAME, SEG_TYPE, SEG_TYPE_NAME,         TABLE_SPACE,
ROW_ID,         USERNAME, OS_USERNAME, MACHINE_NAME, AUDIT_SESSIONID,
SESSION#, SERIAL#, SESSION_INFO,         THREAD#, SEQUENCE#, RBASQN, RBABLK,
RBABYTE,         UBAFIL, UBABLK, UBAREC,  UBASQN,         ABS_FILE#,REL_FILE#,
DATA_BLK#,         DATA_OBJ#, DATA_OBJV#, DATA_OBJD#,         SQL_REDO,
SQL_UNDO, RS_ID, SSN, CSF, INFO,         STATUS, REDO_VALUE, UNDO_VALUE,
SAFE_RESUME_SCN, CSCN, OBJECT_ID         from x$logmnr_contents where ROW_TYPE
= 0


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$LOGMNR_DBA_OBJECTS
select u.inst_id, u.name, o.name, o.subname, o.obj#, o.dataobj#,
decode(o.type#, 0, 'NEXT OBJECT', 1, 'INDEX', 2, 'TABLE', 3, 'CLUSTER',
4, 'VIEW', 5, 'SYNONYM', 6, 'SEQUENCE',                   7, 'PROCEDURE', 8,
'FUNCTION', 9, 'PACKAGE',                  11, 'PACKAGE BODY', 12, 'TRIGGER',
13, 'TYPE', 14, 'TYPE BODY',                  19, 'TABLE PARTITION', 20,
'INDEX PARTITION', 21, 'LOB',                     22, 'LIBRARY', 23,
'DIRECTORY', 24, 'QUEUE',                  28, 'JAVA SOURCE', 29, 'JAVA
CLASS', 30, 'JAVA RESOURCE',                      32, 'INDEXTYPE', 33,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
```

```
'OPERATOR',                    34, 'TABLE SUBPARTITION', 35, 'INDEX
SUBPARTITION',                 40, 'LOB PARTITION', 41, 'LOB SUBPARTITION',
42, 'MATERIALIZED VIEW',               43, 'DIMENSION',
44, 'CONTEXT', 46, 'RULE SET', 47, 'RESOURCE PLAN',                48,
'CONSUMER GROUP',              51, 'SUBSCRIPTION', 52, 'LOCATION',
55, 'XML SCHEMA', 56, 'JAVA DATA',                57, 'SECURITY PROFILE',
59, 'RULE',                60, 'CAPTURE', 61, 'APPLY',
62, 'EVALUATION CONTEXT',              66, 'JOB', 67, 'PROGRAM', 68, 'JOB
CLASS', 69, 'WINDOW',                  72, 'WINDOW GROUP', 74, 'SCHEDULE',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
79, 'CHAIN',                   81, 'FILE GROUP',
'UNDEFINED'),      o.ctime, o.mtime,      to_char(o.stime,
'YYYY-MM-DD:HH24:MI:SS'),       decode(o.status, 0, 'N/A', 1, 'VALID',
'INVALID'),       decode(bitand(o.flags, 2), 0, 'N', 2, 'Y', 'N'),
decode(bitand(o.flags, 4), 0, 'N', 4, 'Y', 'N'),       decode(bitand(o.flags,
16), 0, 'N', 16, 'Y', 'N') from x$logmnr_obj$ o, x$logmnr_user$ u where o.owner#
= u.user#   and o.linkname is null   and (o.type# not in (1  /* INDEX - handled
below */,                  10 /* NON-EXISTENT */)       or       (o.type#
= 1 and 1 = (select 1                          from x$logmnr_ind$ i

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
where i.obj# = o.obj#                          and i.type# in (1, 2, 3, 4,
6, 7, 9))))   and o.name != '_NEXT_OBJECT'   and o.name !=
'_default_auditing_options_'

GV$LOGMNR_DICTIONARY
select INST_ID, DB_NAME, DB_ID, DB_CREATED, TIMESTAMP, RESET_SCN,
RESET_SCN_TIME, DB_VERSION_TIME, DB_CHARACTER_SET,               DB_VERSION,
DB_STATUS, DICTIONARY_SCN, ENABLED_THREAD_MAP,               DB_TXN_SCN,
FILENAME, INFO, STATUS       from x$logmnr_dictionary

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

GV$LOGMNR_DICTIONARY_LOAD
select inst_id, session_id, logmnr_uid,                    action#,
opcode, command, current_state,              completed_actions,
total_actions, loaded, percent_done         from x$logmnr_dictionary_load

GV$LOGMNR_EXTENTS
select e.inst_id, ds.owner, ds.segment_name, ds.partition_name,
ds.segment_type, ds.tablespace_name,       e.ext#, f.file#, e.block#, e.length
```

VIEW_NAME

VIEW_DEFINITION
----
* ds.blocksize, e.length, e.file# from x$logmnr_uet$ e, v$logmnr_sys_dba_segs
ds, x$logmnr_file$ f where e.segfile# = ds.relative_fno and e.segblock# =
ds.header_block and e.ts# = ds.tablespace_id and ds.tablespace_id = f.ts#
and e.file# = f.relfile# and bitand(NVL(ds.segment_flags,0), 1) = 0 and
bitand(NVL(ds.segment_flags,0), 65536) = 0 union all select e.inst_id, ds.owner,
ds.segment_name, ds.partition_name, ds.segment_type, ds.tablespace_name,
e.ktfbueextno, f.file#, e.ktfbuebno, e.ktfbueblks * ds.blocksize,
e.ktfbueblks, e.ktfbuefno from v$logmnr_sys_dba_segs ds, x$logmnr_ktfbue e,
x$logmnr_file$ f where e.ktfbuesegfno = ds.relative_fno and e.ktfbuesegbno =

VIEW_NAME

VIEW_DEFINITION
----
ds.header_block and e.ktfbuesegtsn = ds.tablespace_id and e.ktfbuesegtsn =
f.ts# and e.ktfbuefno = f.relfile# and bitand(NVL(ds.segment_flags, 0), 1) =
1 and bitand(NVL(ds.segment_flags,0), 65536) = 0

GV$LOGMNR_LATCH
select inst_id, session_id, name, child_addr, decode(state, 0,
'UNINIT', 1, 'READY') state from x$logmnr_latch

GV$LOGMNR_LOGFILE

VIEW_NAME

VIEW_DEFINITION
----
select inst_id, log_id, filename, low_time, next_time, db_id,
db_name, reset_scnwrp, reset_scnbas, reset_scn_time, thread_id,
thread_sqn, low_scnwrp, low_scnbas, next_scnwrp, next_scnbas,
decode(state, 0, 'FILE_NOT_OPEN', 1, 'FILE_OPEN', 2, 'DONE') from
x$logmnr_logfile

GV$LOGMNR_LOGS
select INST_ID, LOG_ID, FILENAME, LOW_TIME, HIGH_TIME, DB_ID,
DB_NAME, RESET_SCN, RESET_SCN_TIME, THREAD_ID, THREAD_SQN,

VIEW_NAME

VIEW_DEFINITION

LOW_SCN, NEXT_SCN, DICTIONARY_BEGIN, DICTIONARY_END, TYPE,
BLOCKSIZE, FILESIZE, INFO, STATUS from x$logmnr_logs

GV$LOGMNR_OBJECT_SEGMENTS
select userenv('instance'), s.OWNER, s.SEGMENT_NAME, s.PARTITION_NAME,
s.SEGMENT_TYPE, s.SEGMENT_TYPE_ID, s.TABLESPACE_ID, s.TABLESPACE_NAME,

s.BLOCKSIZE, s.HEADER_FILE, s.HEADER_BLOCK, s.BYTES, s.BLOCKS,
s.EXTENTS, s.INITIAL_EXTENT, s.NEXT_EXTENT, s.MIN_EXTENTS,
s.MAX_EXTENTS, s.PCT_INCREASE, s.FREELISTS, s.FREELIST_GROUPS,

VIEW_NAME

VIEW_DEFINITION
---
s.RELATIVE_FNO, s.BUFFER_POOL_ID, s.SEGMENT_FLAGS, s.SEGMENT_OBJD from
v$logmnr_sys_dba_segs s, v$logmnr_dba_objects o where s.segment_name =
o.object_name and s.segment_type = o.object_type

GV$LOGMNR_PARAMETERS
select INST_ID, START_DATE, REQUIRED_START_DATE, END_DATE, START_SCN,
REQUIRED_START_SCN, END_SCN, OPTIONS, INFO, STATUS from
x$logmnr_parameters

VIEW_NAME

VIEW_DEFINITION

GV$LOGMNR_PROCESS
select a.inst_id, a.session_id, a.pid, a.role, a.work_microsec,
a.overhead_microsec, b.spid, b.username, b.latchwait,
b.latchspin, c.sid, c.serial# from x$logmnr_process a,
v$process b, v$session c where a.pid = b.pid and b.spid = c.process

GV$LOGMNR_REGION
select inst_id, memstate, decode(state, 0, 'INIT', 1, 'AVAIL',
2, 'ASSIGNED', 3, 'PREPARED', 4, 'ASSEMBLED'),

VIEW_NAME

VIEW_DEFINITION
---
owning_process from x$logmnr_region

GV$LOGMNR_SESSION
select INST_ID, session_id, session_name, decode(state, 1,
'READY', 2, 'STARTED', 3, 'ACTIVE', 4,
'DISCARDED', 5, 'DETACHED', 'UNKNOWN'), db_name, db_id,
reset_scn, reset_timestamp, num_process, chunk_size,
start_scn, end_scn, spill_scn, processed_scn, prepared_scn,
read_scn, low_mark_scn, consumed_scn,

VIEW_NAME

VIEW_DEFINITION
---
max_memory_size, used_memory_size, builder_work_size,
prepared_work_size, available_work_size, available_txn,

available_committed_txn, delivered_txn, delivered_committed_txn,
pinned_txn, pinned_committed_txn, checkpoint_interval from
x$logmnr_session

GV$LOGMNR_STATS
select inst_id, session_id, name, value from x$krvxsv where bitand(flags, 1) = 1

VIEW_NAME

VIEW_DEFINITION

GV$LOGMNR_SYS_DBA_SEGS
select u.inst_id, NVL(u.name, 'SYS'), o.name, o.subname, so.object_type,
s.type#, ts.ts#, ts.name, ts.blocksize, f.file#, s.block#,
s.blocks * ts.blocksize, s.blocks, s.extents, s.iniexts * ts.blocksize,
decode(bitand(ts.flags, 3), 1, to_number(NULL),
s.extsize * ts.blocksize), s.minexts, s.maxexts,
decode(bitand(ts.flags, 3), 1, to_number(NULL), s.extpct),
decode(bitand(ts.flags, 32), 32, to_number(NULL), decode(s.lists,
0, 1, s.lists)), decode(bitand(ts.flags, 32), 32, to_number(NULL),

VIEW_NAME

VIEW_DEFINITION
---
decode(s.groups, 0, 1, s.groups)), s.file#, s.cachehint, NVL(s.spare1,0),
o.dataobj# from x$logmnr_user$ u, x$logmnr_obj$ o, x$logmnr_ts$ ts,
v$logmnr_sys_objects so, x$logmnr_seg$ s, x$logmnr_file$ f where s.file# =
so.header_file and s.block# = so.header_block and s.ts# = so.ts_number and
s.ts# = ts.ts# and o.obj# = so.object_id and o.owner# = u.user# (+) and
s.type# = so.segment_type_id and o.type# = so.object_type_id and s.ts# =
f.ts# and s.file# = f.relfile# union all select u.inst_id, NVL(u.name, 'SYS'),
un.name, NULL, decode(s.type#, 1, 'ROLLBACK', 10, 'TYPE2 UNDO'), s.type#,
ts.ts#, ts.name, ts.blocksize, f.file#, s.block#, s.blocks *

VIEW_NAME

VIEW_DEFINITION
---
ts.blocksize, s.blocks, s.extents, s.iniexts * ts.blocksize, s.extsize *
ts.blocksize, s.minexts, s.maxexts, s.extpct,
decode(bitand(ts.flags, 32), 32, to_number(NULL), decode(s.lists, 0, 1,
s.lists)), decode(bitand(ts.flags, 32), 32, to_number(NULL),
decode(s.groups, 0, 1, s.groups)), s.file#, s.cachehint, NVL(s.spare1,0),
un.us# from x$logmnr_user$ u, x$logmnr_ts$ ts, x$logmnr_undo$ un, x$logmnr_seg$
s, x$logmnr_file$ f where s.file# = un.file# and s.block# = un.block#
and s.ts# = un.ts# and s.ts# = ts.ts# and s.user# = u.user# (+) and
s.type# in (1, 10) and un.status$!= 1 and un.ts# = f.ts# and un.file# =

VIEW_NAME

```
VIEW_DEFINITION
--------------------------------------------------------------------------------
f.relfile# union all select u.inst_id,        NVL(u.name, 'SYS'),
to_char(f.file#) || '.' || to_char(s.block#), NULL,       decode(s.type#, 2,
'DEFERRED ROLLBACK', 3, 'TEMPORARY',                      4, 'CACHE', 9, 'SPACE
HEADER', 'UNDEFINED'), s.type#,       ts.ts#, ts.name, ts.blocksize,
f.file#, s.block#,        s.blocks * ts.blocksize, s.blocks, s.extents,
s.iniexts * ts.blocksize,        decode(bitand(ts.flags, 3), 1, to_number(NULL),
s.extsize * ts.blocksize),        s.minexts, s.maxexts,
decode(bitand(ts.flags, 3), 1, to_number(NULL),
s.extpct),        decode(bitand(ts.flags, 32), 32, to_number(NULL),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(s.lists, 0, 1, s.lists)),        decode(bitand(ts.flags, 32), 32,
to_number(NULL),         decode(s.groups, 0, 1, s.groups)),        s.file#,
s.cachehint, NVL(s.spare1,0), s.hwmincr from x$logmnr_user$ u, x$logmnr_ts$ ts,
x$logmnr_seg$ s, x$logmnr_file$ f where s.ts# = ts.ts#   and s.user# = u.user#
(+)   and s.type# not in (1, 5, 6, 8, 10)   and s.ts# = f.ts#   and s.file# =
f.relfile#

GV$LOGMNR_SYS_OBJECTS
select inst_id,         decode(bitand(t.property, 8192), 8192, 'NESTED TABLE',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
'TABLE'), 2, 5,        t.obj#, t.file#, t.block#, t.ts# from x$logmnr_tab$ t
where bitand(t.property, 1024) = 0          /* exclude clustered tables */
union all select inst_id, 'TABLE PARTITION', 19, 5,        tp.obj#, tp.file#,
tp.block#, tp.ts# from x$logmnr_tabpart$ tp union all select inst_id, 'CLUSTER',
3, 5,        c.obj#, c.file#, c.block#, c.ts# from x$logmnr_clu$ c union all
select inst_id, decode(i.type#, 8, 'LOBINDEX', 'INDEX'), 1, 6,        i.obj#,
i.file#, i.block#, i.ts# from x$logmnr_ind$ i where i.type# in (1, 2, 3, 4, 6,
7, 8, 9) union all select inst_id, 'INDEX PARTITION', 20, 6,        ip.obj#,
ip.file#, ip.block#, ip.ts# from x$logmnr_indpart$ ip union all select inst_id,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
'LOBSEGMENT', 21, 8,        l.lobj#, l.file#, l.block#, l.ts# from x$logmnr_lob$
l where (bitand(l.property, 64) = 0) or       (bitand(l.property, 128) = 128)
union all select inst_id, 'TABLE SUBPARTITION', 34, 5,        tsp.obj#,
tsp.file#, tsp.block#, tsp.ts#        from x$logmnr_tabsubpart$ tsp union all
select inst_id, 'INDEX SUBPARTITION', 35, 6,        isp.obj#, isp.file#,
isp.block#, isp.ts# from x$logmnr_indsubpart$ isp union all select inst_id,
decode(lf.fragtype$, 'P', 'LOB PARTITION', 'LOB SUBPARTITION'),
decode(lf.fragtype$, 'P', 40, 41), 8,        lf.fragobj#, lf.file#, lf.block#,
```

lf.ts# from x$logmnr_lobfrag$ lf

VIEW_NAME

VIEW_DEFINITION

GV$LOGMNR_TRANSACTION
select inst_id, session_id, xid, xidusn, xidslt, xidsqn,
parentxid, parent_xidusn, parent_xidslt, parent_xidsqn,
starttimestamp, startscn, chunk, totalchunks, redothread, lowts,
lowscn, committimestamp, commitscn, numchangerecord,
numlcrspilled, dflag, mflag, mflag2, state, type, mining_status,
queue from x$krvxtx

VIEW_NAME

VIEW_DEFINITION

GV$LOGSTDBY
select inst_id,serial#,logstdby_id,pid,type,status_code,status,high_scn from
x$krvslv where exists (select 1 from v$session s, x$knstacr x where
s.sid=x.sid_knst and s.serial#=x.serial_knst)

GV$LOGSTDBY_PROCESS
select inst_id, sid, serial#,logstdby_id,pid,type,status_code,status,high_scn
from x$krvslv where exists (select 1 from v$session s, x$knstacr x where
s.sid=x.sid_knst and s.serial#=x.serial_knst)

VIEW_NAME

VIEW_DEFINITION

GV$LOGSTDBY_PROGRESS
select inst_id,applied_scn,applied_time,restart_scn,restart_time,latest_scn,
latest_time,mining_scn, mining_time from x$krvslvpg

GV$LOGSTDBY_STATE
select inst_id, primary_dbid,session_id,realtime_apply,state from x$krvslvst

GV$LOGSTDBY_STATS

VIEW_NAME

VIEW_DEFINITION
---
select inst_id, name,value from (select inst_id, name,value from x$krvslvs union
all select inst_id, name,to_char(value) from (select inst_id, session_id, name,
value from x$krvxsv where bitand(flags, 2) = 2)where session_id = (select value
from x$krvslvs where name = 'logminer session id')) where exists (select 1 from

v$session s, x$knstacr x where s.sid=x.sid_knst and s.serial#=x.serial_knst)

GV$LOGSTDBY_TRANSACTION
select lt.inst_id, lt.xidusn, lt.xidslt, lt.xidsqn, lt.xid,
lt.startscn, lt.starttimestamp, lt.parent_xidusn,

VIEW_NAME

VIEW_DEFINITION
---
lt.parent_xidslt, lt.parent_xidsqn, lt.parentxid, lt.type,
lt.mining_status, case when sas.sid is null then 'NONE' else
'ACTIVE' end, sas.sid, sas.serial# from x$krvslvs ls,
x$krvxtx lt, v$streams_apply_server sas where (ls.name =
'logminer session id' and ls.value = lt.session_id) and
(lt.chunk = 0 and (lt.xidusn != 0 and lt.xidslt != 0 and
lt.xidsqn != 0)) and (lt.xidusn = sas.xidusn(+) and lt.xidslt =
sas.xidslt(+) and lt.xidsqn = sas.xidsqn(+))

VIEW_NAME

VIEW_DEFINITION

GV$LOG_HISTORY
select
x$kcclh.inst_id,lhrid,lhstm,lhthp,lhseq,to_number(lhlos),to_date(lhlot,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'), to_number(lhnxs), to_number(lhrls),
to_date(lhrlc,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian') from x$kcclh

GV$MANAGED_STANDBY
select inst_id, decode(MSTYP, 1,'RFS', 2,'MRP0', 3,'MR(fg)', 4,'ARCH',
5,'FGRD', 6,'LGWR', 7,'RFS(FAL)', 8,'RFS(NEXP)', 9,'LNS',

VIEW_NAME

VIEW_DEFINITION

'UNKNOWN'), to_number(MSPID), decode(MSSTS, 0,'UNUSED', 1,'ALLOCATED',
2,'CONNECTED', 3,'ATTACHED', 4,'IDLE', 5,'ERROR', 6,'OPENING',
7,'CLOSING', 8,'WRITING', 9,'RECEIVING', 10,'ANNOUNCING',
11,'REGISTERING', 12,'WAIT_FOR_LOG', 13,'WAIT_FOR_GAP', 14,'APPLYING_LOG',
'UNKNOWN'), decode(MSPAR, 0,'N/A', 4,'ARCH', 5,'Archival', 6,'LGWR',
9,'LNS', 'UNKNOWN'), decode(MSPPID, 0,'N/A', to_number(MSPPID)),
decode(MSDBID, 0,'N/A', to_number(MSDBID)), decode(MSLNO, 0,'N/A',
to_number(MSLNO)), to_number(MSRLC), to_number(MSTHR), to_number(MSSEQ),
to_number(MSBNO), to_number(MSBCT), to_number(MSDLY), to_number(MSRCLT),

VIEW_NAME

VIEW_DEFINITION
---

to_number(MSACLT) from x$krssms

GV$MAP_COMP_LIST
select inst_id,elem_idx,num_comp,comp1_name,
comp1_val,comp2_name,comp2_val,comp3_name,
comp3_val,comp4_name,comp4_val,comp5_name,comp5_val from x$ksfmcompl

GV$MAP_ELEMENT
select inst_id,elem_name,elem_idx,elem_cfgid,

VIEW_NAME

VIEW_DEFINITION
----
decode(elem_type,1,'MIRROR',2,'STRIPE',3,'RAID5',
4,'CONCATENATED',5,'PARTITION',6,'DISK',7,'NONE'),
to_number(decode(elem_size,4294967295,NULL,elem_size)),
elem_nsubelem,elem_descr,stripe_size,
to_number(decode(lib_idx,4294967295,NULL,lib_idx)) from x$ksfmelem

GV$MAP_EXT_ELEMENT
select inst_id,elem_idx,num_attrb,attrb1_name,
attrb1_val,attrb2_name,attrb2_val,attrb3_name,

VIEW_NAME

VIEW_DEFINITION
----
attrb3_val,attrb4_name,attrb4_val,attrb5_name,attrb5_val from x$ksfmextelem

GV$MAP_FILE
select inst_id,file_idx,file_cfgid,decode(file_status,1, 'VALID',2,'INVALID'
),file_name,decode(file_type,1,'DATAFILE',
2,'SPFILE',3,'TEMPFILE',4,'CONTROLFILE',5,'LOGFILE',6,'ARCHIVEFILE'),
decode(file_struct,1,'FILE',2,'RAWVOLUME',3,'RAWDEVICE',4,'NONE'),
file_size,file_nexts,to_number(decode(lib_idx,4294967295,NULL,lib_idx)) from
x$ksfmfile

VIEW_NAME

VIEW_DEFINITION

GV$MAP_FILE_EXTENT
select inst_id,file_idx,ext_num,ext_dev_off,
ext_size,ext_file_off,decode(ext_type,1,'DATA', 2,'PARITY',3,'NONE'),elem_idx
from x$ksfmfileext where elem_idx != 4294967295

GV$MAP_FILE_IO_STACK
select inst_id,file_idx,depth,elem_idx,cu_size,stride,num_cu,
dev_offset,to_number(decode(file_offset,4294967295,NULL,file_offset)),

VIEW_NAME

VIEW_DEFINITION
---
decode(data_type,1,'DATA',2,'PARITY',3,'DATA AND PARITY'),parity_pos,
parity_perd,row_id,prow_id from x$ksfmiost

GV$MAP_LIBRARY
select inst_id,lib_idx,lib_name,vendor_name,protocol_num,
version_num,path_name,decode(bitand(cap_file,1),0,'N',1,'Y'),
decode(bitand(cap_file, 6),0,'NONE',6,'PERSISTENT',2,'NONPERSISTENT'),
decode(bitand(cap_elem, 1),0,'N',1,'Y'),decode(bitand(cap_elem,6),0,'NONE',
6,'PERSISTENT',4,'NONPERSISTENT'),decode(cap_other,0,'N',1,'Y') from x$ksfmlib

VIEW_NAME

VIEW_DEFINITION

GV$MAP_SUBELEMENT
select inst_id,child_idx,parent_idx,sub_num,
to_number(decode(sub_size,4294967295,NULL,sub_size)),
to_number(decode(elem_offset,4294967295,NULL,elem_offset)),sub_flags from
x$ksfmsubelem where child_idx != 4294967295

GV$MAX_ACTIVE_SESS_TARGET_MTH
select inst_id, policy_name_kgskasp from x$kgskasp

VIEW_NAME

VIEW_DEFINITION

GV$MEMORY_CURRENT_RESIZE_OPS
select sc.inst_id, sc.component, decode(sc.opcode, 0, 'STATIC', 1,
'INITIALIZING', 2, 'DISABLED', 3, 'GROW', 4, 'SHRINK', 5,
'SHRINK_CANCEL', NULL), decode(sc.opmode, 1, 'MANUAL', 2, 'DEFERRED', 3,
'IMMEDIATE', NULL), pn.name, sc.initsize * sc.gransize, sc.targsize *
sc.gransize, sc.cursize * sc.gransize, sc.starttime, sc.lasttime from
x$kmgsct sc, v$parameter pn where (sc.parno = pn.num) and (sc.opcode <> 0)
and (sc.starttime is not null)

VIEW_NAME

VIEW_DEFINITION

GV$MEMORY_DYNAMIC_COMPONENTS
select st.inst_id, st.component, st.cursize * st.gransize, st.minsize *
st.gransize, st.maxsize * st.gransize, st.usersize * st.gransize,
st.opercnt, decode(st.lastoper, 0, 'STATIC', 1, 'INITIALIZING', 2,
'DISABLED', 3, 'GROW', 4, 'SHRINK', 5, 'SHRINK_CANCEL', NULL),

decode(st.lastmode, 1, 'MANUAL', 2, 'DEFERRED', 3, 'IMMEDIATE', NULL),
st.lasttime, st.gransize from x$kmgsct st

VIEW_NAME

VIEW_DEFINITION

GV$MEMORY_RESIZE_OPS
select op.inst_id, gv.component, decode(op.opcode, 0, 'STATIC', 1,
'INITIALIZING', 2, 'DISABLED', 3, 'GROW', 4, 'SHRINK', 5,
'SHRINK_CANCEL', NULL), decode(op.opmode, 1, 'MANUAL', 2, 'DEFERRED', 3,
'IMMEDIATE', NULL), pn.name, op.initsize * gv.gransize, op.targsize *
gv.gransize, op.realsize * gv.gransize, decode(op.status, 0,
'INACTIVE', 1, 'PENDING', 2, 'COMPLETE', 3, 'CANCELLED', 4, 'ERROR',
5, 'ERROR', 6, 'CANCELLED', 7, 'CANCELLED', NULL), op.starttime,
op.endtime from x$kmgsop op, x$kmgsct gv, v$parameter pn where (op.grantype

VIEW_NAME

VIEW_DEFINITION

= gv.grantype) and (op.parno = pn.num) order by op.starttime

GV$MEMORY_TARGET_ADVICE
select A.inst_id, A.memsz,
round((A.memsz / A.base_memsz), 4),
decode(A.base_estd_dbtime,0,to_number(null),
round(A.base_dbtime * round((A.dbtime / A.base_estd_dbtime), 4), 0)),
decode(A.base_estd_dbtime,0,to_number(null),
round((A.dbtime / A.base_estd_dbtime), 4)),

VIEW_NAME

VIEW_DEFINITION

A.version from
x$kmgsbsmemadv A order
by A.inst_id

GV$METRIC
SELECT inst_id, begtime, endtime, intsize_csec, groupid, eid, eidsq,
metricid, name, value, unit FROM x$kewmdrmv WHERE flag1 = 1

GV$METRICGROUP

VIEW_NAME

VIEW_DEFINITION

SELECT inst_id, groupid, name, intsize, maxintv FROM x$kewmgsm

GV$METRICNAME
SELECT d.inst_id, d.groupid, g.name, d.metricid, d.name, d.unit
FROM x$kewmdsm d, x$kewmgsm g where d.groupid = g.groupid

GV$METRIC_HISTORY
SELECT inst_id, begtime, endtime, intsize_csec, groupid, eid, eidsq,
metricid, name, value, unit FROM x$kewmdrmv

VIEW_NAME

VIEW_DEFINITION

GV$MTTR_TARGET_ADVICE
select distinct inst_id, mttr_v,
decode(status, 0, 'OFF', 4, 'ON', 'READY'),
decode(dirty_limit, 0, to_number(NULL), dirty_limit),
decode(factored_sim_writes, -1, to_number(NULL), factored_sim_writes),
decode(base_real_nondirect_writes, 0, to_number(NULL),
decode(factored_sim_writes, -1, to_number(NULL),
round((factored_sim_writes / base_real_nondirect_writes),

VIEW_NAME

VIEW_DEFINITION
---
4))), decode(total_writes, -1,
to_number(NULL), total_writes), decode(base_total_writes, 0,
to_number(NULL), decode(total_writes, -1,
to_number(NULL), round((total_writes /
base_total_writes), 4))), decode(total_ios, -1, to_number(NULL),
total_ios), decode(base_total_ios, 0, to_number(NULL),
decode(total_ios, -1, to_number(NULL),
round((total_ios / base_total_ios), 4))) from x$kcbmmav

VIEW_NAME

VIEW_DEFINITION

GV$MUTEX_SLEEP
select INST_ID, MUTEX_TYPE, LOCATION, SLEEPS, WAIT_TIME from x$MUTEX_SLEEP

GV$MUTEX_SLEEP_HISTORY
select INST_ID, MUTEX_IDENTIFIER, SLEEP_TIMESTAMP, MUTEX_TYPE, GETS, SLEEPS,
REQUESTING_SESSION, BLOCKING_SESSION, LOCATION, MUTEX_VALUE, P1, P1RAW, P2, P3,
P4, P5 from x$MUTEX_SLEEP_HISTORY

GV$MVREFRESH

VIEW_NAME

VIEW_DEFINITION
---
select inst_id, sid_knst, serial_knst, currmvowner_knstmvr, currmvname_knstmvr
from x$knstmvr x where type_knst=6 and exists (select 1 from v$session s where
s.sid=x.sid_knst and s.serial#=x.serial_knst)

GV$MYSTAT
select inst_id,ksusenum,ksusestn,ksusestv from x$ksumysta where
bitand(ksspaflg,1)!=0 and bitand(ksuseflg,1)!=0 and ksusestn<(select ksusgstl
from x$ksusgif)

VIEW_NAME

VIEW_DEFINITION

GV$NFS_CLIENTS
SELECT inst_id, NFSCLIENTID, NFSPRINCIPAL, NFSOPAQUECLIENT,
NFSVERIFIER, NFSLEASEEXPIRY, NFSCLIENTNETID || NFSCLIENTADDR,
decode(bitand(NFSFLAGS,1), 1, 'TRUE', 'FALSE') FROM X$NFSCLIENTS

GV$NFS_LOCKS
SELECT inst_id, NFSOPENSTATEID, NFSOPENSEQNO, NFSLOCKSTATEID,
NFSLOCKSEQNO, NFSLOCKOWNER, NFSLOCKOFFSET, NFSLOCKLENGTH, NFSLOCKTYPE
FROM X$NFSLOCKS

VIEW_NAME

VIEW_DEFINITION

GV$NFS_OPEN_FILES
SELECT inst_id, NFSCLIENTID, NFSOPENOWNER, NFSOPENSTATEID,
NFSOPENFILEHANDLE, NFSOPENSEQID, decode(bitand(NFSOPENFLAGS,1), 0,
'FALSE', 'TRUE'), decode(bitand(NFSOPENFLAGS,2), 0, 'FALSE', 'TRUE'),
decode(bitand(NFSOPENFLAGS,384), 384, 'SharedReadWrite', 128, 'SharedRead', 256,
'SharedWrite'), decode(bitand(NFSOPENFLAGS,1536), 1536,
'DenyReadWrite', 512, 'DenyRead', 1024, 'DenyWrite'),
decode(bitand(NFSOPENFLAGS,64), 0, 'FALSE', 'TRUE') FROM X$NFSOPENS

VIEW_NAME

VIEW_DEFINITION

GV$NLS_PARAMETERS
select inst_id,parameter, value from x$nls_parameters where parameter !=
'NLS_SPECIAL_CHARS'

GV$NLS_VALID_VALUES
select inst_id,parameter, value, decode (isdeprecated, 0, 'FALSE', 'TRUE') from
x$ksulv

VIEW_NAME

VIEW_DEFINITION

GV$OBJECT_DEPENDENCY
select d.inst_id,d.kglhdpar, d.kglnahsh, o.kglnaown, o.kglnaobj,
o.kglhdadr, o.kglnahsh, o.kglobtyp from x$kglob o, x$kgldp d where o.kglnahsh =
d.kglrfhsh and o.kglhdadr = d.kglrfhdl

GV$OBSOLETE_PARAMETER
select inst_id,kspponm,decode(ksppoval,0,'FALSE','TRUE') from x$ksppo

GV$OFFLINE_RANGE

VIEW_NAME

VIEW_DEFINITION
---
select x$kccor.inst_id,orrid,orstm,ordfp,to_number(orofs),to_number(orons),
to_date(oront,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),
to_number(orrls), to_date(orrlc,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian')
from x$kccor

GV$OPEN_CURSOR
select inst_id,kgllkuse, kgllksnm, user_name, kglhdpar, kglnahsh,
kgllksqlid, kglnaobj, kgllkest,
decode(kgllkexc, 0, to_number(NULL), kgllkexc) from

VIEW_NAME

VIEW_DEFINITION
---
x$kgllk where kglhdnsp = 0 and kglhdpar != kgllkhdl

GV$OPTION
select inst_id,parameter, value from x$option

GV$OSSTAT
select INST_ID, KSUCPUSTATNAME, decode(KSUCPUSTATID,
15, KSUCPUSTATVALUE/1024, KSUCPUSTATVALUE), KSUCPUSTATID,
KSUCPUSTATCMT, decode(bitand(KSUCPUSTATFLAGS,1),1,'YES','NO') from

VIEW_NAME

VIEW_DEFINITION

X$KSUCPUSTAT union all select INST_ID, KSUVMSTATNAME,
KSUVMSTATVALUE, KSUVMSTATID, KSUVMSTATCMT,
decode(bitand(KSUVMSTATFLAGS,1),1,'YES','NO') from X$KSUVMSTAT union all
select INST_ID, KSUNETSTATNAME, KSUNETSTATVALUE,

```
KSUNETSTATID,       KSUNETSTATCMT,
decode(bitand(KSUNETSTATFLAGS,1),1,'YES','NO')   from X$KSUNETSTAT

GV$PARALLEL_DEGREE_LIMIT_MTH
select inst_id, policy_name_kgskdopp from x$kgskdopp

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

GV$PARAMETER
select x.inst_id,x.indx+1,ksppinm,ksppity,ksppstvl, ksppstdvl, ksppstdf,
decode(bitand(ksppiflg/256,1),1,'TRUE','FALSE'),
decode(bitand(ksppiflg/65536,3),1,'IMMEDIATE',2,'DEFERRED',
3,'IMMEDIATE','FALSE'),  decode(bitand(ksppiflg,4),4,'FALSE',
decode(bitand(ksppiflg/65536,3), 0, 'FALSE', 'TRUE')),
decode(bitand(ksppstvf,7),1,'MODIFIED',4,'SYSTEM_MOD','FALSE'),
decode(bitand(ksppstvf,2),2,'TRUE','FALSE'),  decode(bitand(ksppilrmflg/64, 1),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
1, 'TRUE', 'FALSE'),  decode(bitand(ksppilrmflg/268435456, 1), 1, 'TRUE',
'FALSE'),  ksppdesc, ksppstcmnt, ksppihash  from x$ksppi x, x$ksppcv y where
(x.indx = y.indx) and  ((translate(ksppinm,'_','#') not like '###%') and
((translate(ksppinm,'_','#') not like '#%')      or (ksppstdf = 'FALSE') or
(bitand(ksppstvf,5) > 0)))

GV$PARAMETER2
select x.inst_id,kspftctxpn,ksppinm,ksppity,kspftctxvl,  kspftctxdvl,
kspftctxdf,  decode(bitand(ksppiflg/256,1),1,'TRUE','FALSE'),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(bitand(ksppiflg/65536,3),1,'IMMEDIATE',2,'DEFERRED',
3,'IMMEDIATE','FALSE'),  decode(bitand(ksppiflg,4),4,'FALSE',
decode(bitand(ksppiflg/65536,3), 0, 'FALSE', 'TRUE')),
decode(bitand(kspftctxvf,7),1,'MODIFIED',4,'SYSTEM_MOD','FALSE'),
decode(bitand(kspftctxvf,2),2,'TRUE','FALSE'),  decode(bitand(ksppilrmflg/64,
1), 1, 'TRUE', 'FALSE'),  decode(bitand(ksppilrmflg/268435456, 1), 1, 'TRUE',
'FALSE'),  ksppdesc, kspftctxvn, kspftctxct  from x$ksppi x, x$ksppcv2 y  where
((x.indx+1) = kspftctxpn) and  ((translate(ksppinm,'_','#') not like '###%') and
((translate(ksppinm,'_','#') not like '#%')      or (kspftctxdf = 'FALSE') or

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
```

(bitand(kspftctxvf,5) > 0)))

GV$PARAMETER_VALID_VALUES
SELECT INST_ID, PARNO_KSPVLD_VALUES, NAME_KSPVLD_VALUES, ORDINAL_KSPVLD_VALUES,
VALUE_KSPVLD_VALUES, ISDEFAULT_KSPVLD_VALUES FROM X$KSPVLD_VALUES WHERE
TRANSLATE(NAME_KSPVLD_VALUES,'_','#') NOT LIKE '#%'

GV$PERSISTENT_PUBLISHERS
select p.inst_id, p.queue_id, q.queue_schema, q.queue_name, p.publisher_name,

VIEW_NAME

VIEW_DEFINITION
----
p.publisher_address, p.protocol, p.enqueued_msgs, p.elapsed_enqueue_time,
p.last_enqueue_time from x$persistent_publishers p, x$persistent_queues q
where p.inst_id = q.inst_id and p.queue_id = q.queue_id

GV$PERSISTENT_QUEUES
select inst_id, queue_id, queue_schema, queue_name, first_activity_time,
enqueued_msgs, dequeued_msgs, browsed_msgs, elapsed_enqueue_time,
elapsed_dequeue_time, elapsed_transformation_time,
elapsed_rule_evaluation_time, enqueued_expiry_msgs, enqueued_delay_msgs,

VIEW_NAME

VIEW_DEFINITION
----
msgs_made_expired, msgs_made_ready, last_enqueue_time, last_dequeue_time,
last_tm_expiry_time, last_tm_ready_time from x$persistent_queues

GV$PERSISTENT_SUBSCRIBERS
select s.inst_id, s.queue_id, q.queue_schema, q.queue_name, s.subscriber_id,
s.subscriber_name, s.subscriber_address, s.protocol, s.subscriber_type,
s.first_activity_time, s.enqueued_msgs, s.dequeued_msgs, s.browsed_msgs,
s.expired_msgs, s.dequeued_msg_latency, s.last_enqueue_time, s.last_dequeue_time
from x$persistent_subscribers s, x$persistent_queues q where s.inst_id =

VIEW_NAME

VIEW_DEFINITION
----
q.inst_id and s.queue_id = q.queue_id

GV$PGASTAT
select INST_ID,   QESMMSGANM,   decode(QESMMSGAUN, 3,
(QESMMSGAVL*QESMMSGAMU)/100,   QESMMSGAVL*QESMMSGAMU),
decode(QESMMSGAUN, 0, 'bytes', 1, 'microseconds', 3, 'percent', '')   from
X$QESMMSGA   where QESMMSGAVS = 1

GV$PGA_TARGET_ADVICE

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select INST_ID,         PAT_PRED * 1024,         round(PAT_PRED/PAT_CURR,
4),         decode(status, 0, 'OFF', 'ON'),         BYTES_PROCESSED * 1024,
ESTD_TIME,                         EXTRA_BYTES_RW * 1024,
round(decode(BYTES_PROCESSED+EXTRA_BYTES_RW, 0, 0,
(BYTES_PROCESSED*100)/(BYTES_PROCESSED+EXTRA_BYTES_RW))),         OVERALLOC
from   X$QESMMAPADV

GV$PGA_TARGET_ADVICE_HISTOGRAM
select INST_ID,         PAT_PRED * 1024,         round(PAT_PRED/PAT_CURR,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
4),         decode(status, 0, 'OFF', 'ON'),         LOWBND * 1024,
(HIBND * 1024)-1,         OPTIMAL,         ONEPASS,         MPASS,
MPASS+ONEPASS+OPTIMAL,         IGNORED    from   X$QESMMAHIST

GV$PQ_SESSTAT
select inst_id, kxfpssnam, kxfpssval, kxfpsstot from x$kxfpsst

GV$PQ_SLAVE
select inst_id,kxfpdpnam, decode(bitand(kxfpdpflg, 16), 0, 'BUSY', 'IDLE'),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
kxfpdpses, floor(kxfpdpcit / 6000), floor(kxfpdpcbt / 6000),  floor(kxfpdpcct /
100), kxfpdpclsnt + kxfpdpcrsnt,  kxfpdpclrcv + kxfpdpcrrcv,  floor((kxfpdptit +
kxfpdpcit) / 6000),  floor((kxfpdptbt + kxfpdpcbt) / 6000),  floor((kxfpdptct +
kxfpdpcct) / 100),  kxfpdptlsnt + kxfpdpclsnt + kxfpdptrsnt + kxfpdpcrsnt,
kxfpdptlrcv + kxfpdpclrcv + kxfpdptrrcv + kxfpdpcrrcv  from x$kxfpdp where
bitand(kxfpdpflg, 8) != 0

GV$PQ_SYSSTAT
select inst_id, rpad(kxfpysnam,30), kxfpysval from x$kxfpys

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

GV$PQ_TQSTAT
select inst_id, kxfqsqn, kxfqsid, rpad(kxfqsty,10), kxfqscnt, kxfqslen, kxfqset,
kxfqsavl, kxfqsdw, kxfqsdt, rpad(kxfqssid,10), kxfqsiid from x$kxfqsrow

GV$PROCESS
```

```
select inst_id, addr,indx,ksuprpid,ksuprunm,ksuprser,ksuprtid,ksuprpnm,
ksuprtfi, ksuprtfn, decode(bitand(ksuprflg,2),0,null,1),
decode(ksllawat,hextoraw('00'),null,ksllawat),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(ksllaspn,hextoraw('00'),null,ksllaspn),
ksuprpum,ksuprpnam+ksuprpram,ksuprpfm, case when ksuprpnam+ksuprpram > ksuprpmm
then ksuprpnam+ksuprpram       else ksuprpmm end  from x$ksupr where
bitand(ksspaflg,1)!=0

GV$PROCESS_GROUP
SELECT indx, inst_id, ksupgpnm, ksupgpid FROM x$ksupgp

GV$PROCESS_MEMORY

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select inst_id, ksmpgst_pid, ksmpgst_ser, ksmpgst_catname, ksmpgst_alloc,
decode(ksmpgst_used, 0, to_number(null), 1, 0, ksmpgst_used),
decode(ksmpgst_maxal, 0, to_number(null), 1, 0, ksmpgst_maxal)  from x$ksmpgst
where bitand(ksmpgst_paflg,1)!=0 and        (ksmpgst_alloc > 0 or ksmpgst_used
> 1 or ksmpgst_maxal > 1)        order by inst_id, ksmpgst_pid, ksmpgst_catidx

GV$PROCESS_MEMORY_DETAIL
select inst_id, ksmpgdst_pid, ksmpgdst_ser, ksmpgdst_catname,  ksmpgdst_comment,
ksmpgdst_heapname, ksmpgdst_bytes_alloc,  ksmpgdst_num_alloc, ksmpgdst_ds,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
ksmpgdst_parent_ds  from x$ksmpgdst where bitand(ksmpgdst_paflg,1)!=0 and
(ksmpgdst_bytes_alloc > 0)        order by inst_id, ksmpgdst_pid,
ksmpgdst_bytes_alloc

GV$PROCESS_MEMORY_DETAIL_PROG
select inst_id, ksmpgdp_pid, ksmpgdp_ser, ksmpgdp_status  from x$ksmpgdp where
bitand(ksmpgdp_paflg,1)!=0        order by inst_id, ksmpgdp_pid

GV$PROPAGATION_RECEIVER

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select inst_id, kwqpdsqs, kwqpdsqn, kwqpddbn,
decode(bitand(max(kwqpdflg), 1), 1, max(kwqpddqs), null),
```

```
decode(bitand(max(kwqpdflg), 1), 1, max(kwqpddqn), null),       min(kwqpdtim),
decode(bitand(max(kwqpdflg), 2), 0, max(kwqpdhwm), 0),       min(kwqpdack),
max(kwqpdhwm), max(kwqpdmsg), min(kwqpdupc),       min(kwqpdrul),
min(kwqpdenq) - min(kwqpdupc) - min(kwqpdrul)  from x$kwqpd group by inst_id,
kwqpdsqs, kwqpdsqn, kwqpdddbn
```

GV$PROPAGATION_SENDER

VIEW_NAME

VIEW_DEFINITION

```
select s.inst_id, s.kwqpsqid, q.queue_schema, q.queue_name,       s.kwqpsdqs,
s.kwqpsdqn, s.kwqpstim,       s.kwqpsdbn, s.kwqpshwm, s.kwqpsack, s.kwqpsstt,
s.kwqpsmsg, s.kwqpsbyt, s.kwqpsdeq, s.kwqpspic,       s.kwqpsprp - s.kwqpsdeq
- s.kwqpspic,       s.kwqpsmmsg, s.kwqpsmbyt, s.kwqpslmsglat ,
s.kwqpslmsgenqtime,       CAST(s.kwqpslmsgenqtime + s.kwqpslmsglat/86400 AS
TIMESTAMP),       s.kwqpsllcrlat, s.kwqpsllcrts,       s.kwqpsllcrts +
s.kwqpsllcrlat/86400, s.kwqpsddbnm  from x$kwqps s, x$buffered_queues q  where
s.inst_id = q.inst_id and s.kwqpsqid = q.queue_id  and bitand(q.flags, 16) = 0
```

VIEW_NAME

VIEW_DEFINITION

GV$PROXY_ARCHIVEDLOG

```
select
inst_id,pcrid,pcstm,pcdev,pchdl,pccmt,pcmdh,pcmpl,pctag,decode(bitand(pcflg,
1+2+4),0,'A',1,'D',2,'X',4,'U','?'),decode(bitand(pcflg,1),1,'YES','NO'),pathp,p
aseq,to_number(parls),to_date(parlc,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(palos),to_date(palot,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(panxs),to_date(panxt,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),pabct,pabsz,to_date(pctsm,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_date(pctim,'MM/DD/RR
```

VIEW_NAME

VIEW_DEFINITION

```
HH24:MI:SS','NLS_CALENDAR=Gregorian'),abs((to_date(pctim,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian')- to_date(pctsm,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'))*86400), pcrsi, pcrst,
decode(bitand(paflg, 1), 1, 'YES', 'NO'), decode(bitand(paflg, 1792), 0, 'NO',
'YES'),to_date(pakpt,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(bitand(paflg, 1792), 256,  'LOGS',
512,  'NOLOGS',                1024, 'BACKUP_LOGS',
NULL)from x$kccpa
```

VIEW_NAME

```
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$PROXY_DATAFILE
select
inst_id,pcrid,pcstm,pcdev,pchdl,pccmt,pcmdh,pcmpl,pctag,decode(bitand(pcflg,
1+2+4),0,'A',1,'D',2,'X',4,'U','?'),decode(bitand(pcflg,1),1,'YES','NO'),pddfp,t
o_number(pdcrs),to_date(pdcrt,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(pdrls),to_date(pdrlc,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(pdcps),to_date(pdcpt,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_number(pdafs),to_number(pdrfs),to_date(
pdrft,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(bitand(pdflg,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
1),1,0,NULL),decode(bitand(pdflg, 2),0,'NO','YES'),decode(bitand(pdflg,
4),0,'NO','YES'),pdfsz,pdbsz,pdlor,to_date(pctsm,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),to_date(pctim,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),abs((to_date(pctim,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian')- to_date(pctsm,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'))*86400), decode(pddfp, 0,
decode(bitand(pcflg,8),8,'S','B'), NULL), decode(bitand(pdflg, 1792), 0, 'NO',
'YES'),         to_date(pdkpt,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),decode(bitand(pdflg, 1792), 256,  'LOGS',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
512,  'NOLOGS',                         1024, 'BACKUP_LOGS',
NULL),       pcrsi, pcrst, pcfdi, decode(bitand(pdflg, 16), 0, 'NO', 'YES'),
pcplus, pcprls, pcprlt from x$kccpd

GV$PWFILE_USERS
select inst_id,username,decode(sysdba,1,'TRUE','FALSE'),
decode(sysoper,1,'TRUE','FALSE'), decode(sysasm,1,'TRUE','FALSE')  from x$kzsrt
where valid=1  and username != 'INTERNAL'


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$PX_BUFFER_ADVICE
select inst_id, rpad(kxfpnsnam,30), kxfpnsval from x$kxfpns where indx = 4  or
indx >= 14

GV$PX_INSTANCE_GROUP
select pig.inst_id, kxfppigpig, decode(kxfppigsrc, 1, 'PARALLEL_INSTANCE_GROUP',
2, 'SERVICE', NULL), kxfpiginst from x$kxfppig pig, x$kxfpig ig  where
pig.kxfppigpig = ig.kxfpigig
```

VIEW_NAME

VIEW_DEFINITION

GV$PX_PROCESS
select a.inst_id, a.kxfpdpnam, decode(bitand(a.kxfpdpflg, 16), 0, 'IN USE',
'AVAILABLE'), b.pid, a.kxfpdpspid, c.sid, c.serial# from x$kxfpdp a, V$PROCESS
b, V$SESSION c where bitand(kxfpdpflg, 8) != 0 and a.kxfpdpspid = b.SPID and
a.kxfpdpspid = c.PROCESS(+)

GV$PX_PROCESS_SYSSTAT
select inst_id, rpad(kxfpnsnam,30), kxfpnsval from x$kxfpns where indx < 15

VIEW_NAME

VIEW_DEFINITION

GV$PX_SESSION
select a.inst_id, a.addr, a.indx, a.ksuseser,
decode(b.kxfpdpqcsid,NULL,a.indx,b.kxfpdpqcsid), b.kxfpdpqcser, b.kxfpdpcin,
b.kxfpdpsvgrp, b.kxfpdpsvset, b.kxfpdpsvnum, b.kxfpdpadg, b.kxfpdprdg from
x$ksuse a, x$kxfpdp b where bitand(a.ksspaflg,1)!=0 and
bitand(a.ksuseflg,1)!=0 and a.ksuseqcsid > 0 and a.ksusepro = b.kxfpdppro(+)

GV$PX_SESSTAT
select a.inst_id, a.addr, a.indx, a.ksuseser,

VIEW_NAME

VIEW_DEFINITION
---
decode(b.kxfpdpqcsid,NULL,a.indx,b.kxfpdpqcsid), b.kxfpdpqcser, b.kxfpdpcin,
b.kxfpdpsvgrp, b.kxfpdpsvset, b.kxfpdpsvnum, b.kxfpdpadg, b.kxfpdprdg,
c.ksusestn, c.ksusestv from x$ksuse a, x$kxfpdp b, x$ksusesta c where
bitand(a.ksspaflg,1)!=0 and bitand(a.ksuseflg,1)!=0 and a.KSUSEQCSID > 0 and
a.ksusepro = b.kxfpdppro(+) and a.indx = c.ksusenum and c.ksusestn < (select
ksusgstl from x$ksusgif)

GV$QUEUE
select inst_id,kmcqspro,decode(bitand(kmcqstyp,1),1,'COMMON','DISPATCHER'),

VIEW_NAME

VIEW_DEFINITION
---
kmcqsncq,kmcqswat,kmcqstnc from x$kmcqs where bitand(kmcqsflg,1) = 1 and
bitand(kmcqstyp,3) != 0 and (bitand(kmcqstyp,8) = 8 or kmcqspro !=
hextoraw('00') or kmcqstnc > 0)

```
GV$QUEUEING_MTH
select inst_id, policy_name_kgskquep from x$kgskquep

GV$RECOVERY_FILE_STATUS
select fn.inst_id, fn.fnfno, fn.fnnam, decode(nvl(mf.cps, 0), 0, 'NOT

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
RECOVERED', 281474976710655, 'CURRENT', 'IN RECOVERY') from x$kcrmx mx, x$kccfn
fn, x$kccfe fe, x$kcrmf mf where fn.fntyp = 4 and mf.fno(+) = fn.fnfno and
((bitand(mx.flg,2) != 0 and fe.fedup != 0) or mf.fno = fn.fnfno) and fe.fenum =
fn.fnfno

GV$RECOVERY_LOG
select inst_id,lhthp,lhseq, to_date(lhlot,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),  nvl(lhnam, alnam)  from x$kcclh,
(select althp, alseq, alrls, alrlc, alnam /* filter out duplicate al */

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
from (select althp, alseq, alrls, alrlc, alnam, alstm, alrid,
max(alstm) over /* get latest archivelog */                    (partition
by althp, alseq, alrls, alrlc) almstm,                  max(alrid) over  /*
filter out duplicate stamp */                    (partition by althp,
alseq, alrls, alrlc,                                alstm) almrid
from x$kccal             where bitand(alflg, 8) = 0    /* standby_dest = NO
*/              and bitand(alflg, 2) = 2    /* archived = YES */
and bitand(alflg, 1) = 0)   /* deleted = NO */        where alrid = almrid
and alstm = almstm)  where lhthp = althp (+)    and lhseq = alseq (+)    and

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
lhrls = alrls (+)    and lhrlc = alrlc (+)    and to_number(lhnxs) > (select
min(to_number(fhscn)) from x$kcvfhmrr where hxerr = 0) and lhseq not in (select
leseq from x$kccle where lethr = lhthp) and to_number(lhlos) < (select
max(to_number(hxsts)) from x$kcvfhmrr where hxerr = 0)

GV$RECOVERY_PROGRESS
select inst_id,   to_date(ksulostm,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),   ksulopna, ksulotde, ksulouni, ksulosfr,
ksulotot,   to_date(ksuloinft, 'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian')

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
```

```
from x$ksulop where ksulopna like '% Recovery' order by 1, 2 DESC, 5 ASC, 4 ASC

GV$RECOVERY_STATUS
select fx.inst_id, to_date(mx.ckptim,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'), mx.thr, mx.seq, mx.los,
to_date(mx.tim,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'), nvl(mx.nam,
'NONE'), decode(bitand(mx.mrs, 256 + 128 + 64 + 8), 8, 'RELEASE', 64, 'WRONG
LOG', 128, 'MISSING NAME', 256, 'UNNEEDED NAME', 'NONE'), decode(nvl(fx.err, 3),
1, 'NEED LOG', 3, 'END OF THREAD', 4, 'LOG REUSED', 'UNKNOWN') from x$kcrmx mx,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
x$kcrfx fx where fx.thr(+) = mx.thr

GV$RECOVER_FILE
select inst_id,hxfil, decode(hxons, 0, 'OFFLINE', 'ONLINE'),decode(hxons, 0,
'OFFLINE', 'ONLINE'), decode(hxerr, 0,'',1,'FILE MISSING', 2,'OFFLINE NORMAL',
3,'NOT VERIFIED', 4,'FILE NOT FOUND',5,'CANNOT OPEN FILE', 6,'CANNOT READ
HEADER', 7,'CORRUPT HEADER',8,'WRONG FILE TYPE', 9,'WRONG DATABASE', 10,'WRONG
FILE NUMBER',11,'WRONG FILE CREATE', 12,'WRONG FILE CREATE', 16,'DELAYED
OPEN','UNKNOWN ERROR'), to_number(fhscn),  to_date(fhtim,'MM/DD/RR

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
HH24:MI:SS','NLS_CALENDAR=Gregorian') from x$kcvfhmrr

GV$REDO_DEST_RESP_HISTOGRAM
select inst_id,  DEST_ID,  TIME,  DURATION,  FREQUENCY  from x$kcrrnhg

GV$REPLPROP
select inst_id, sid_knst, serial_knst, decode(type_knst, 3, 'Replication
Parallel Prop Slave'|| slavenum_knstrpp, 4, 'Replication Parallel Prop
Coordinator'), dblink_knstrpp, decode(state_knstrpp, 0, NULL, 1, 'WAIT', 2,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
'SLEEP', 3, 'PUSH', 4, 'PURGE' , 5, 'CREATE ERROR', 6, 'SCHEDULE TXN'),
decode(type_knst, 4, NULL, xid_knstrpp), sequence_knstrpp from x$knstrpp x where
type_knst in (3,4) and exists (select 1 from v$session s where s.sid=x.sid_knst
and s.serial#=x.serial_knst)

GV$REPLQUEUE
select inst_id, txns_enqueued_knstrqu, calls_enqueued_knstrqu,
txns_purged_knstrqu, last_enqueue_time_knstrqu,last_purge_time_knstrqu  from
x$knstrqu
```

VIEW_NAME

VIEW_DEFINITION

GV$REQDIST
select inst_id,kmmrdbuc,sum(kmmrdcnt) from x$kmmrd where
kmmrdpro!=hextoraw('00') group by inst_id,kmmrdbuc

GV$RESERVED_WORDS
select inst_id, keyword, length,
decode(mod(trunc(type/2),2),0,'N',1,'Y','?') reserved,
decode(mod(trunc(type/4),2),0,'N',1,'Y','?') res_type,

VIEW_NAME

VIEW_DEFINITION
---
decode(mod(trunc(type/8),2),0,'N',1,'Y','?') res_attr,
decode(mod(trunc(type/16),2),0,'N',1,'Y','?') res_semi,
decode(mod(trunc(type/32),2),0,'N',1,'Y','?') duplicate from x$kwddef

GV$RESOURCE
select inst_id,addr,ksqrsidt,ksqrsid1,ksqrsid2 from x$ksqrs where
bitand(ksqrsflg,2)!=0

GV$RESOURCE_LIMIT

VIEW_NAME

VIEW_DEFINITION
---
select inst_id, ksurlmnm, ksurlmcv, ksurlmmv, LPAD(decode(bitand(ksurlmfg, 1),
0, to_char(ksurlmia), 'UNLIMITED'),10), LPAD(decode(bitand(ksurlmfg, 2), 0,
to_char(ksurlmlv), 'UNLIMITED'),10) from x$ksurlmt

GV$RESTORE_POINT
select rsp.inst_id, to_number(rsp.rspscn), rsp.rspincarn,
decode(bitand(rsp.rspflags, 1), 1, 'YES', 'NO'),
to_number(rsp.rsplgsz), to_timestamp(rsp.rsptime,
'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),

VIEW_NAME

VIEW_DEFINITION
---
to_timestamp(decode(bitand(rsp.rspflags, 8), 0, NULL,
rsp.rsprsptime), 'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'), 'YES',
rsp.rspname from x$kccrsp rsp where bitand(rsp.rspflags, 2)
!= 0 union all select rsp.inst_id,
to_number(rsp.nrsscn), rsp.nrsincarn, 'NO', 0,

```
to_timestamp(rsp.nrstime,                    'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),
to_timestamp(decode(bitand(rsp.nrsflags, 8), 0, NULL,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
rsp.nrsrsptime),                     'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),          'NO',
rsp.nrsname         from  x$kccnrs rsp         where bitand(rsp.nrsflags, 2)
!= 0

GV$RESULT_CACHE_DEPENDENCY
select  INST_ID,          QESRCDEP_RID,       QESRCDEP_DID,
QESRCDEP_OBJ from x$qesrcdep


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$RESULT_CACHE_MEMORY
select  INST_ID,          QESRCMEM_LAD,       QESRCMEM_CNK,
QESRCMEM_BLK,       decode(QESRCMEM_STA, 0, 'NO', 1, 'YES'),
QESRCMEM_OID,       QESRCMEM_POS from x$qesrcmem

GV$RESULT_CACHE_OBJECTS
select  INST_ID,            QESRCOBJ_MEM_LAD, decode(QESRCOBJ_CAC_TYP,
1,'Result', 2,'Dependency','Other'), decode(QESRCOBJ_RSE_STA,
1,decode(QESRCOBJ_RSE_BYP,0,'New','Bypass'),2,'Published',3,'Invalid',4,'Expired

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
','Other'),       QESRCOBJ_CAC_BUC,       QESRCOBJ_CAC_HSV,
QESRCOBJ_CAC_NAM, decode(QESRCOBJ_CAC_NSP, 0, 'SQL', 1, 'PLSQL', 2, 'API', 3,
'AUTO'),       QESRCOBJ_CAC_DAT,       QESRCOBJ_CAC_UID,
QESRCOBJ_DEP_DCT,       QESRCOBJ_RSE_BCT,       QESRCOBJ_CAC_SCN,
QESRCOBJ_RSE_CCT,       QESRCOBJ_RSE_PCT,       QESRCOBJ_RSE_SCT,
QESRCOBJ_RSE_RCT,       QESRCOBJ_RSE_RMX,       QESRCOBJ_RSE_RMN,
QESRCOBJ_RSE_RAG,       QESRCOBJ_RSE_ETM,       QESRCOBJ_RSE_POS,
QESRCOBJ_DEP_OBJ,       QESRCOBJ_DEP_INV,       QESRCOBJ_RSE_SPO,
QESRCOBJ_RSE_SPU,       QESRCOBJ_CAC_CID,       QESRCOBJ_CAC_KEY from

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
x$qesrcobj
```

GV$RESULT_CACHE_STATISTICS
select INST_ID, INDX + 1, QESRCSTA_NAM, QESRCSTA_VAL
from x$qesrcsta

GV$RESUMABLE
select inst_id, ktrsfaddr, ktrsfsid, decode (bitand(ktrsfflg, 1), 0, 'NO',
'YES'), decode (ktrsfsta, 0, 'NORMAL', 1, 'SUSPENDED', 2, 'TIMEOUT', 3, 'ERROR',

VIEW_NAME

VIEW_DEFINITION

4, 'ABORTED', ''), ktrsftmo, ktrsfspt, ktrsfrst, ktrsfnam, ktrsferr, ktrsfep1,
ktrsfep2, ktrsfep3, ktrsfep4, ktrsfep5, ktrsfems from x$ktrso

GV$RFS_THREAD
select INST_ID, THREAD#, RESETLOG_SCN, RESETLOG#, LAST_REDO_SEQ#,
LAST_REDO_BLK#, LAST_REDO_TIME, LOW_GAP_SCN, LOW_GAP_TIME, LAST_PING_TIME FROM
x$krfsthrd

GV$RMAN_COMPRESSION_ALGORITHM

VIEW_NAME

VIEW_DEFINITION

SELECT inst_id, id_krbmca, algname_krbmca, algdesc_krbmca, algvsn_krbmca,
kckrcl_krbmca, DECODE(isvalid_krbmca, 0, 'NO', 'YES'), DECODE(isdefault_krbmca,
0, 'NO', 'YES') FROM x$krbmca

GV$RMAN_CONFIGURATION
select INST_ID, RMRNO, RMNAM, RMVAL from X$KCCRM where RMNAM is not null

GV$RMAN_ENCRYPTION_ALGORITHMS
select inst_id, id, algname, algdesc, decode(isdefault, 0, 'NO', 'YES'),

VIEW_NAME

VIEW_DEFINITION
---
decode(restore_only, 0, 'NO', 'YES') from x$krbza

GV$RMAN_OUTPUT
select userenv('Instance'),SID_KRBMROT, ROWNO_KRBMROT, MTS_KRBMROT,
L0ID_KRBMROT,L0TS_KRBMROT,TXT_KRBMROT, ID_KRBMROT, STAMP_KRBMROT,
L0ID_KRBMROT
SESSION_KEY from x$krbmrot order by L0ID_KRBMROT, L0TS_KRBMROT,
MTS_KRBMROT,ROWNO_KRBMROT

GV$RMAN_STATUS_CURRENT

VIEW_NAME

```
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select userenv('Instance'), SID_KRBMRST, ID_KRBMRST, STAMP_KRBMRST,
decode(LEVEL_KRBMRST, 0, to_number(NULL), PID_KRBMRST),
decode(LEVEL_KRBMRST, 0, to_number(NULL), PTS_KRBMRST),       L0ID_KRBMRST,
L0TS_KRBMRST, LEVEL_KRBMRST,        decode(LEVEL_KRBMRST, 0, 'SESSION',
1, 'COMMAND',                        'RECURSIVE OPERATION'),
CMDID_KRBMRST, UPPER(OPER_KRBMRST),        decode(STATUS_KRBMRST, 1, 'RUNNING',
1+8, 'RUNNING WITH WARNINGS',                1+16, 'RUNNING WITH
ERRORS',                  1+8+16, 'RUNNING WITH ERRORS',
2, 'COMPLETED',                      2+8, 'COMPLETED WITH WARNINGS',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
2+16, 'COMPLETED WITH ERRORS',                  2+8+16, 'COMPLETED WITH
ERRORS',                        'FAILED'),        0,
to_date(START_KRBMRST,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),
to_date(END_KRBMRST,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian')  from
x$krbmrst

GV$ROLLSTAT
select inst_id,kturdusn,kturdlat,kturdext,kturdsiz,kturdwrt,kturdnax,
kturdget,kturdwat,decode(kturdopt, -1,to_number(null),kturdopt),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
kturdhwm,kturdnsh,kturdnwp,kturdnex,kturdash,kturdaae,
decode(bitand(kturdflg,127), 0,'ONLINE', 2,'PENDING OFFLINE', 3,'OFFLINE',
4, 'FULL', 'UNKNOWN'), kturdcex, kturdcbk from x$kturd where kturdsiz != 0 and
bitand(kturdflg,127) != 3

GV$ROWCACHE
select inst_id,kqrstcid,decode(kqrsttyp,1,'PARENT','SUBORDINATE'),
decode(kqrsttyp,2,kqrstsno,null),kqrsttxt,kqrstcsz,kqrstusg,kqrstfcs,
kqrstgrq,kqrstgmi,kqrstsrq,kqrstsmi,kqrstsco,kqrstmrq,kqrstmfl,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
kqrstilr,kqrstifr,kqrstisr from x$kqrst

GV$ROWCACHE_PARENT
select inst_id, indx, kqrfphsh, kqrfpadd, kqrfpcid, kqrfpcnm,
decode(bitand(kqrfpflg, 1), 0, 'Y', 'N'),  kqrfpmod, kqrfpreq, kqrfptxn,
kqrfpses, kqrfpirq, kqrfpirl,  kqrfpity, kqrfpii1, kqrfpii2, kqrfpkey from
x$kqrfp
```

GV$ROWCACHE_SUBORDINATE

VIEW_NAME

VIEW_DEFINITION
---
select inst_id, indx, kqrfshsh, kqrfsadd, kqrfscid, kqrfssid, kqrfssnm,
decode(bitand(kqrfsflg, 1), 0, 'Y', 'N'), kqrfspar, kqrfskey from x$kqrfs

GV$RSRCMGRMETRIC
SELECT m.inst_id, m.begtime, m.endtime, m.intsize_csec,
m.sequence#, c.consumer_group_id_kgskscs, c.name_kgskscs,
m.cpu_consumed_time, m.cpu_wait_time, m.io_requests,
m.io_megabytes FROM x$kewmrmgmv m, x$kskplw p, x$kgskscs c
WHERE flag1 = 1 AND m.sequence# = p.seq_kskplw AND

VIEW_NAME

VIEW_DEFINITION
---
p.end_tm_kskplw is null AND p.seq_kskplw = c.seq_kgskscs
AND m.consumer_group_id = c.num_kgskscs

GV$RSRCMGRMETRIC_HISTORY
SELECT m.inst_id, m.begtime, m.endtime, m.intsize_csec,
m.sequence#, c.consumer_group_id_kgskscs, c.name_kgskscs,
m.cpu_consumed_time, m.cpu_wait_time, m.io_requests,
m.io_megabytes FROM x$kewmrmgmv m, x$kskplw p, x$kgskscs c
WHERE m.sequence# = p.seq_kskplw AND p.end_tm_kskplw is null

VIEW_NAME

VIEW_DEFINITION

AND p.seq_kskplw = c.seq_kgskscs AND m.consumer_group_id =
c.num_kgskscs

GV$RSRC_CONSUMER_GROUP
select A.inst_id, A.name_kgskcft, A.class_id_kgskcft,
A.current_count_kgskcft, A.runnable_count_kgskcft,
A.total_count_kgskcft, A.cpu_wait_kgskcft, A.cpu_waits_kgskcft,
A.total_used_kgskcft, A.yields_kgskcft,
A.num_queued_kgskcft,

VIEW_NAME

VIEW_DEFINITION

A.undo_consump_kgskcft, A.active_limit_hit_kgskcft,
A.undo_limit_hit_kgskcft,
A.swch_in_time_kgskcft, A.swch_out_time_kgskcft,

```
A.swch_in_mb_kgskcft, A.swch_out_mb_kgskcft,
A.swch_in_reqs_kgskcft, A.swch_out_reqs_kgskcft,
A.call_aborted_kgskcft, A.actv_sess_killed_kgskcft,
A.idle_sess_killed_kgskcft, A.idlblkr_sess_kld_kgskcft,
A.queued_time_kgskcft, A.queue_timeouts_kgskcft,
B.wtime_ksfdstcg, B.wreqs_ksfdstcg,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
round(B.sbrdata_ksfdstcg / 2048),
round(B.sbwdata_ksfdstcg / 2048),
round(B.mbrdata_ksfdstcg / 2048),
round(B.mbwdata_ksfdstcg / 2048),
B.sbrreqs_ksfdstcg, B.sbwreqs_ksfdstcg,
B.mbrreqs_ksfdstcg, B.mbwreqs_ksfdstcg                          from
x$kgskcft A, x$ksfdstcg B                                  where
A.class_id_kgskcft = B.consumer_group_id_ksfdstcg              and
B.pool_ksfdstcg = 1

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

GV$RSRC_CONSUMER_GROUP_CPU_MTH
select inst_id, policy_name_kgskcp from x$kgskcp

GV$RSRC_CONS_GROUP_HISTORY
select inst_id,        seq_kgskscs,        consumer_group_id_kgskscs,
name_kgskscs,        requests_kgskscs,        cpu_wait_time_kgskscs,
cpu_waits_kgskscs,        consumed_cpu_time_kgskscs,        yields_kgskscs,
active_sess_limit_hit_kgskscs,        undo_limit_hit_kgskscs,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
swch_in_time_kgskscs, swch_out_time_kgskscs,        swch_in_mb_kgskscs,
swch_out_mb_kgskscs,        swch_in_reqs_kgskscs, swch_out_reqs_kgskscs,
sql_canceled_kgskscs,        active_sess_killed_kgskscs,
idle_sess_killed_kgskscs,        idle_blkr_sess_killed_kgskscs,
queued_time_kgskscs,        queue_time_outs_kgskscs,        wtime_kgskscs,
wreqs_kgskscs,        round(sbrdata_kgskscs / 2048), round(sbwdata_kgskscs /
2048),        round(mbrdata_kgskscs / 2048), round(mbwdata_kgskscs / 2048),
sbrreqs_kgskscs, sbwreqs_kgskscs,        mbrreqs_kgskscs, mbwreqs_kgskscs
from x$kgskscs

VIEW_NAME
------------------------------
VIEW_DEFINITION
```

```
--------------------------------------------------------------------------------

GV$RSRC_PLAN
select inst_id, plan_id_kgskpft, name_kgskpft,
decode(is_top_plan_kgskpft, 0, 'FALSE', 'TRUE'),        decode(cpu_rm_kgskpft,
0, 'OFF', 'ON')         from x$kgskpft

GV$RSRC_PLAN_CPU_MTH
select inst_id, policy_name_kgskpp from x$kgskpp


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$RSRC_PLAN_HISTORY
select inst_id,         plw.seq_kskplw,         decode(plw.id_kskplw, 0,
to_number(null), plw.id_kskplw),         plw.name_kskplw,
plw.start_tm_kskplw,         plw.end_tm_kskplw,
decode(plw.window_id_kskplw, 0, 'FALSE', 'TRUE'),         o.name,
decode(plw.auto_switch_kskplw, 0, 'FALSE', 'TRUE'),
decode(plw.cpu_rm_kskplw, 0, 'OFF', 'ON')         from x$kskplw plw, obj$ o
where plw.window_id_kskplw = o.obj#(+)


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$RSRC_SESSION_INFO
select inst_id, sid_kgskvft,        class_id_kgskvft,
orig_class_id_kgskvft,        orig_class_map_kgskvft, mapped_cg_name_kgskvft,
state_kgskvft,        decode(active_kgskvft, 1, 'TRUE', 0, 'FALSE', ''),
idle_time_kgskvft,        cur_cpu_wait_time_kgskvft, tot_cpu_wait_time_kgskvft,
cur_cpu_waits_kgskvft, tot_cpu_waits_kgskvft,        cur_cpu_time_kgskvft,
tot_cpu_time_kgskvft,        cur_active_time_kgskvft, tot_active_time_kgskvft,
cur_queued_time_kgskvft, tot_queued_time_kgskvft,        cur_yields_kgskvft,
tot_yields_kgskvft,        cur_undo_kgskvft, max_undo_kgskvft,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
call_aborted_kgskvft, queue_timeouts_kgskvft,        est_exec_lmt_hit_kgskvft,
cur_io_service_time_kgskvft, tot_io_service_time_kgskvft,
cur_io_service_waits_kgskvft, tot_io_service_waits_kgskvft,
round(cur_sbrdata_kgskvft / 2048), round(tot_sbrdata_kgskvft / 2048),
round(cur_mbrdata_kgskvft / 2048), round(tot_mbrdata_kgskvft / 2048),
round(cur_sbwdata_kgskvft / 2048), round(tot_sbwdata_kgskvft / 2048),
round(cur_mbwdata_kgskvft / 2048), round(tot_mbwdata_kgskvft / 2048),
cur_sbrreqs_kgskvft, tot_sbrreqs_kgskvft,        cur_sbwreqs_kgskvft,
tot_sbwreqs_kgskvft,        cur_mbrreqs_kgskvft, tot_mbrreqs_kgskvft,
```

VIEW_NAME

VIEW_DEFINITION
---
cur_mbwreqs_kgskvft, tot_mbwreqs_kgskvft from x$kgskvft

GV$RULE
select INST_ID, RULE_SET_OBJECT_ID, EVALUATION_CONTEXT_OBJECT_ID, RULE_OWNER, RULE_NAME, RULE_CONDITION, TRUE_HITS, MAYBE_HITS, SQL_EVALUATIONS from x$rule

GV$RULE_SET
select INST_ID, OWNER, NAME, CPU_TIME, ELAPSED_TIME, FIRST_LOAD_TIME, LAST_LOAD_TIME, LAST_LOADING_TIME, SHARABLE_MEM, RELOADS, INVALIDATIONS,

VIEW_NAME

VIEW_DEFINITION

EVALUATIONS, FIRST_HIT_EVALUATIONS, SIMPLE_RULES_ONLY_EVALUATIONS, SQL_FREE_EVALUATIONS, SQL_EXECUTIONS, CONDITIONS_PROCESSED, TRUE_RULES, MAYBE_RULES, VARIABLE_VALUE_FUNCTION_CALLS, VARIABLE_METHOD_FUNCTION_CALLS, EVALUATION_FUNCTION_CALLS from x$rule_set

GV$RULE_SET_AGGREGATE_STATS
select inst_id, name, value from x$kwrsnv

GV$SCHEDULER_RUNNING_JOBS

VIEW_NAME

VIEW_DEFINITION
---
select inst_id, session_id, session_serial_num, job_id, paddr, os_process_id, CAST (numtodsinterval(session_stat_cpu/100, 'second') AS INTERVAL DAY(3) TO SECOND(2)) session_stat_cpu from x$jskslv

GV$SECUREFILE_TIMER
SELECT INST_ID, KDLU_STATNAME, KDLU_STATLAYERID, KDLU_STATMYTIME, KDLU_STATMAXTIME, KDLU_STATMINTIME, KDLU_STATINVOCATIONS, KDLU_STATLAYERNAME FROM X$KDLU_STAT

VIEW_NAME

VIEW_DEFINITION

GV$SEGMENT_STATISTICS
select s.inst_id, u.name, o.name, o.subname, ts.name, s.fts_tsn, o.obj#, o.dataobj#, decode(o.type#, 0, 'NEXT OBJECT', 1, 'INDEX', 2, 'TABLE', 3, 'CLUSTER', 4, 'VIEW', 5,

```
'SYNONYM', 6, 'SEQUENCE',       7, 'PROCEDURE', 8, 'FUNCTION', 9, 'PACKAGE',
11, 'PACKAGE BODY', 12, 'TRIGGER',           13, 'TYPE', 14, 'TYPE BODY',
19, 'TABLE PARTITION',                    20, 'INDEX PARTITION', 21, 'LOB',
22, 'LIBRARY', 23, 'DIRECTORY', 24, 'QUEUE',  28, 'JAVA SOURCE', 29, 'JAVA

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
CLASS',         30, 'JAVA RESOURCE', 32, 'INDEXTYPE',        33, 'OPERATOR',
34, 'TABLE SUBPARTITION',     35, 'INDEX SUBPARTITION',                  40,
'LOB PARTITION', 41, 'LOB SUBPARTITION',  42, 'MATERIALIZED VIEW',  43,
'DIMENSION',    44, 'CONTEXT', 47, 'RESOURCE PLAN',   48, 'CONSUMER GROUP',
51, 'SUBSCRIPTION', 52, 'LOCATION',   55, 'XML SCHEMA', 56, 'JAVA DATA',    57,
'SECURITY PROFILE',     'UNDEFINED'),     s.fts_statnam,     s.fts_statid,
s.fts_staval       from obj$ o, user$ u,          x$ksolsfts s, ts$ ts  where
o.owner# = u.user#   and s.fts_inte = 0       and s.fts_objn = o.obj#   and
s.fts_tsn = ts.ts#    and s.fts_objd = o.dataobj#     and o.linkname is null

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
and (o.type# not in (1  /* INDEX - handled below */,                   10
/* NON-EXISTENT */)       or                        (o.type# = 1
and 1 = (select 1                   from ind$  i                where
i.obj# = o.obj#                    and i.type# in
(1, 2, 3, 4, 6, 7, 8, 9))))   and o.name != '_NEXT_OBJECT'             and
o.name != '_default_auditing_options_' union all    select       s.inst_id,
u.name,     o.name,     o.subname,   ts.name,     s.fts_tsn,   t.ktssoobjn,
t.ktssoobjd, decode(o.type#, 0, 'NEXT OBJECT', 1, 'INDEX',   2, 'TABLE', 3,
'CLUSTER',                  4, 'VIEW', 5, 'SYNONYM', 6, 'SEQUENCE',       7,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
'PROCEDURE', 8, 'FUNCTION', 9, 'PACKAGE',  11, 'PACKAGE BODY', 12, 'TRIGGER',
13, 'TYPE', 14, 'TYPE BODY',               19, 'TABLE PARTITION',
20, 'INDEX PARTITION', 21, 'LOB',           22, 'LIBRARY', 23, 'DIRECTORY',
24, 'QUEUE',  28, 'JAVA SOURCE', 29, 'JAVA CLASS',         30, 'JAVA RESOURCE',
32, 'INDEXTYPE',        33, 'OPERATOR', 34, 'TABLE SUBPARTITION',     35,
'INDEX SUBPARTITION',                  40, 'LOB PARTITION', 41, 'LOB
SUBPARTITION',  42, 'MATERIALIZED VIEW', 43, 'DIMENSION',     44, 'CONTEXT', 47,
'RESOURCE PLAN',   48, 'CONSUMER GROUP',               51, 'SUBSCRIPTION', 52,
'LOCATION',   55, 'XML SCHEMA', 56, 'JAVA DATA',    57, 'SECURITY PROFILE',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
'UNDEFINED'),       s.fts_statnam,      s.fts_statid,       s.fts_staval
```

```
from obj$ o, user$ u,       x$ksolsfts s, x$ktsso t,       ts$ ts
where o.owner#   = u.user# and          s.fts_inte = 0 and
s.fts_objn = o.obj# and          s.fts_tsn  = t.ktssotsnum and
s.fts_objn = t.ktssoobjn and       s.fts_objd = t.ktssoobjd and
s.fts_tsn  = ts.ts# and          t.ktssotsn = ts.name and
o.linkname is null and         (o.type# not in (1 /* INDEX - handled below */,
10 /* NON-EXISTENT */) or     (o.type# = 1 and 1 = (select 1 from ind$ i where
i.obj# = o.obj# and      i.type# in (1, 2, 3, 4, 6, 7, 8, 9))))    and o.name
```

VIEW_NAME

VIEW_DEFINITION

```
!= '_NEXT_OBJECT'    and o.name != '_default_auditing_options_'
```

GV$SEGSTAT

```
select inst_id,                                fts_tsn,
fts_objn,                             fts_objd,
fts_statnam,                           fts_statid,
fts_staval                          from x$ksolsfts
where fts_inte = 0
```

VIEW_NAME

VIEW_DEFINITION

GV$SEGSTAT_NAME

```
select inst_id,                                st_statid,
st_name,                              decode(bitand(st_flag, 1),
0, 'NO', 1, 'YES')  from x$ksolsstat where bitand(st_flag, 2) <> 2
```

GV$SERVICEMETRIC

```
SELECT sm.inst_id, begtime, endtime, intsize_csec,          groupid,
sv.kswsastabnmh, sv.kswsastabnm, ctmhash,          elapsedpercall, cpupercall,
dbtimepercall, callspersec,          dbtimepersec,  goodness, delta, flags
```

VIEW_NAME

VIEW_DEFINITION

```
FROM   x$kewmsvcmv sm, x$kswsastab sv        WHERE flag1 = 1          AND
sm.svcid = sv.kswsastabsi
```

GV$SERVICEMETRIC_HISTORY

```
SELECT sm.inst_id, begtime, endtime, intsize_csec,          groupid,
sv.kswsastabnmh, sv.kswsastabnm,          ctmhash, elapsedpercall, cpupercall,
dbtimepercall,          callspersec, dbtimepersec        FROM   x$kewmsvcmv
sm, x$kswsastab sv        WHERE sm.svcid = sv.kswsastabsi
```

VIEW_NAME

```
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$SERVICES
select inst_id, kswsastabsi, kswsastabnm, kswsastabnmh, kswsastabnn,
kswsastabcd, kswsastabcdh,  decode(kswsastabgoal, -1, NULL, 0, 'NONE', 1,
'SERVICE_TIME', 2, 'THROUGHPUT', NULL) kswsastabgoal,
decode(bitand(kswsastabpflg, 2), 2, 'Y', 'N') kswsastabpflg,
decode(bitand(kswsastabpflg, 4), 4, 'YES', 'NO'),  decode(bitand(kswsastabpflg,
8), 8, 'LONG', 'SHORT')  from x$kswsastab

GV$SERVICE_EVENT

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select s.inst_id, s.kswsevtabnm, s.kswsevtabnmh, d.ksledname, d.ksledhash,
s.kswsevtabwts, s.kswsevtabtmo, round(s.kswsevtabtim / 10000),
round(s.kswsevtabtim / (10000 * s.kswsevtabwts)),  round(s.kswsevtabmxt /
10000),s.kswsevtabtim from x$kswsevtab s, x$ksled d  where s.kswsevtabwts != 0
and s.kswsevtabnum = d.indx

GV$SERVICE_STATS
select s.inst_id, s.svchsh, s.svcnam, m.extid, m.sname, s.kewsval  from
x$kewssvcv s, x$kewssmap m  where s.kewsoff = m.offst and m.aggid = 3

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

GV$SERVICE_WAIT_CLASS
select s.inst_id, s.kswsclstabnm, s.kswsclstabnmh, s.kswsclsid,  s.kswsclsnum,
s.kswsclsname, s.kswsclswts, round(s.kswsclstim / 10000) from x$kswsclstab s
where s.kswsclswts != 0

GV$SERV_MOD_ACT_STATS
select sma.inst_id, 'SERVICE_MODULE_ACTION', sma.srvnam, sma.modnam,
sma.actnam, m.extid, m.sname, sma.statval  from x$kewesmas sma, x$kewssmap m

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
where sma.statpos = m.offst and m.aggid = 4       union all  select sm.inst_id,
'SERVICE_MODULE', sm.srvnam, sm.modnam, NULL, m.extid,    m.sname, sm.statval
from x$kewesms sm, x$kewssmap m  where sm.statpos = m.offst and m.aggid = 5

GV$SESSION
select
s.inst_id,s.addr,s.indx,s.ksuseser,s.ksuudses,s.ksusepro,s.ksuudlui,s.ksuudlna,s
```

.ksuudoct,s.ksusesow,
decode(s.ksusetrn,hextoraw('00'),null,s.ksusetrn),decode(s.ksqpswat,hextoraw('00

VIEW_NAME

VIEW_DEFINITION

'),null,s.ksqpswat),decode(bitand(s.ksuseidl,11),1,'ACTIVE',0,decode(bitand(s.ks
useflg,4096),0,'INACTIVE','CACHED'),2,'SNIPED',3,'SNIPED',
'KILLED'),decode(s.ksspatyp,1,'DEDICATED',2,'SHARED',3,'PSEUDO','NONE'),
s.ksuudsid,s.ksuudsna,s.ksuseunm,s.ksusepid,s.ksusemnm,s.ksusetid,s.ksusepnm,
decode(bitand(s.ksuseflg,19),17,'BACKGROUND',1,'USER',2,'RECURSIVE','?'),
s.ksusesql, s.ksusesqh, s.ksusesqi, decode(s.ksusesch, 65535, to_number(null),
s.ksusesch), s.ksusesesta, decode(s.ksuseseid, 0, to_number(null),
s.ksuseseid), s.ksusepsq, s.ksusepha, s.ksusepsi, decode(s.ksusepch, 65535,
to_number(null), s.ksusepch), s.ksusepesta, decode(s.ksusepeid, 0,

VIEW_NAME

VIEW_DEFINITION
----
to_number(null), s.ksusepeid), decode(s.ksusepeo,0,to_number(null),s.ksusepeo),
decode(s.ksusepeo,0,to_number(null),s.ksusepes),
decode(s.ksusepco,0,to_number(null),s.ksusepco),
decode(s.ksusepco,0,to_number(null),s.ksusepcs), s.ksuseapp, s.ksuseaph,
s.ksuseact, s.ksuseach, s.ksusecli, s.ksusefix, s.ksuseobj, s.ksusefil,
s.ksuseblk, s.ksuseslt, s.ksuseltm, s.ksusectm,decode(bitand(s.ksusepxopt,
12),0,'NO','YES'),decode(s.ksuseft, 2,'SESSION',
4,'SELECT',8,'TRANSACTIONAL','NONE'),decode(s.ksusefm,1,'BASIC',2,'PRECONNECT',4
,'PREPARSE','NONE'),decode(s.ksusefs, 1, 'YES',

VIEW_NAME

VIEW_DEFINITION

'NO'),s.ksusegrp,decode(bitand(s.ksusepxopt,4),4,'ENABLED',decode(bitand(s.ksuse
pxopt,8),8,'FORCED','DISABLED')),decode(bitand(s.ksusepxopt,2),2,'FORCED',decode
(bitand(s.ksusepxopt,1),1,'DISABLED','ENABLED')),decode(bitand(s.ksusepxopt,32),
32,'FORCED',decode(bitand(s.ksusepxopt,16),16,'DISABLED','ENABLED')),
s.ksusecqd, s.ksuseclid, decode(s.ksuseblocker,4294967295,'UNKNOWN',
4294967294, 'UNKNOWN',4294967293,'UNKNOWN',4294967292,'NO HOLDER',
4294967291,'NOT IN WAIT','VALID'),decode(s.ksuseblocker,
4294967295,to_number(null),4294967294,to_number(null),
4294967293,to_number(null), 4294967292,to_number(null),4294967291,

VIEW_NAME

VIEW_DEFINITION
----
to_number(null),bitand(s.ksuseblocker, 2147418112)/65536),decode(s.ksuseblocker,
4294967295,to_number(null),4294967294,to_number(null),
4294967293,to_number(null), 4294967292,to_number(null),4294967291,

```
to_number(null),bitand(s.ksuseblocker,
65535)),w.kslwtseq,w.kslwtevt,e.ksllednam,e.ksledp1,w.kslwtp1,w.kslwtp1r,
e.ksledp2,w.kslwtp2,w.kslwtp2r,e.ksledp3,w.kslwtp3,w.kslwtp3r,
e.ksledclassid,e.ksledclass#,e.ksledclass, decode(w.kslwtinwait,
0,decode(bitand(w.kslwtflags,256),              0,-2,
decode(round(w.kslwtstime/10000),                    0,-1,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
round(w.kslwtstime/10000))),       0),
decode(w.kslwtinwait,0,round((w.kslwtstime+w.kslwtltime)/1000000),
round(w.kslwtstime/1000000)), decode(w.kslwtinwait,1,'WAITING',
decode(bitand(w.kslwtflags,256),0,'WAITED UNKNOWN TIME',
decode(round(w.kslwtstime/10000),0,'WAITED SHORT TIME',    'WAITED KNOWN
TIME'))),w.kslwtstime, decode(w.kslwtinwait,0,to_number(null),
decode(bitand(w.kslwtflags,64),64,0,w.kslwttrem)), w.kslwtltime,s.ksusesvc,
decode(bitand(s.ksuseflg2,32),32,'ENABLED','DISABLED'),decode(bitand(s.ksuseflg2
,64),64,'TRUE','FALSE'),decode(bitand(s.ksuseflg2,128),128,'TRUE','FALSE'),decod

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
e(bitand(s.ksuseflg2,65536) + bitand(s.ksuseflg2,131072),65536,'ALL
EXEC',131072,'NEVER',0,'FIRST EXEC'),s.ksuudsae,s.ksusecre,s.ksusecsn from
x$ksuse s, x$ksled e, x$kslwt w where bitand(s.ksspaflg,1)!=0 and
bitand(s.ksuseflg,1)!=0 and s.indx=w.kslwtsid and w.kslwtevt=e.indx

GV$SESSION_CONNECT_INFO
select inst_id, ksusenum, ksuseser, decode(ksuseaty, 0, 'DATABASE', 1,  'OS', 2,
'NETWORK', 3, 'PROXY', 4, 'SERVER', 5, 'PASSWORD',  6, 'EXTERNAL ADAPTERS', 7,
'INTERNAL', 8, 'GLOBAL',  9, 'EXTERNAL', 10, 'PASSWORD BASED GLOBAL USER', '?'),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
ksuseunm, ksuseban,  decode(ksusecsid,0,'Unknown',nls_charset_name(ksusecsid)),
decode(bitand(ksuseflags,1), 0, 'Heterogeneous', 'Homogeneous'),
decode(ksusecllib,1,'Home-based',2,'Full Instant Client',
3,'XE Instant Client',4,'Light Weight Instant Client',
5,'OCI','Unknown'),  SYS_OP_VERSION(ksuseclvsn), ksusecldrv,
decode(bitand(ksusecllbf,1), 1, 'Client Temp Lob Rfc On',
'Client Temp Lob Rfc Off'),   ksuseclregid  from x$ksusecon
where bitand(ksuseflg,1)!=0 and bitand(ksuseflg,16)=0


VIEW_NAME
------------------------------
VIEW_DEFINITION
```

```
--------------------------------------------------------------------------------
GV$SESSION_CURSOR_CACHE
select inst_id,kgscugmax,kgscugcnt,kgscugopn,kgscughit,
decode(kgscugopn,0,1,kgscughit/kgscugopn) from x$kgscc

GV$SESSION_EVENT
select s.inst_id, s.kslessid, d.ksllednam, s.ksleswts, s.kslestmo,
round(s.kslestim / 10000), round(s.kslestim / (10000 * s.ksleswts), 2),
round(s.kslesmxt / 10000),  s.kslestim, d.ksledhash, d.ksledclassid,
d.ksledclass#, d.ksledclass  from x$ksles s, x$ksled d  where s.ksleswts != 0

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
and s.kslesenm = d.indx

GV$SESSION_FIX_CONTROL
select  INST_ID,       SID_QKSBGSEROW,       BUGNO_QKSBGSEROW,
VALUE_QKSBGSEROW,       FID_QKSBGSEROW,       DESC_QKSBGSEROW,
OFE_QKSBGSEROW,       EVENT_QKSBGSEROW,       ISDEFAULT_QKSBGSEROW from
x$qksbgses

GV$SESSION_LONGOPS

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select inst_id, ksulosno, ksulosrn, ksulopna,       ksulotna, ksulotde,
ksulosfr, ksulotot, ksulouni,       to_date(ksulostm,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),       to_date(ksulolut,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),       to_date(ksuloinft, 'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),       decode(ksulopna, 'Advisor',
ksuloif2,            decode(sign(ksulotot-ksulosfr),-1,to_number(NULL),
decode(ksulosfr, 0, to_number(NULL),
round(ksuloetm*((ksulotot-ksulosfr)/ksulosfr)))),      ksuloetm, ksuloctx,
ksulomsg, ksulounm,       ksulosql, ksulosqh, ksulosqi, ksulosqph, ksulosqesta,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(ksulosqeid, 0, to_number(NULL), ksulosqeid),       decode(ksulosqplid,
0, to_number(NULL), ksulosqplid),       ksulosqplop, ksulosqplnm, ksuloqid
from x$ksulop

GV$SESSION_OBJECT_CACHE
select
inst_id,kocstpin,kocsthit,kocsttht,decode(kocstpin,0,1,kocsthit/kocstpin),decode
(kocstpin,0,1,kocsttht/kocstpin),kocstorf,kocstrfs,kocstofs,kocstfls,kocstshr,ko
cstcnt,kocstpnd,kocstsiz,kocstopt,kocstmax from x$kocst
```

VIEW_NAME

VIEW_DEFINITION

GV$SESSION_WAIT
select s.inst_id,s.kslwtsid,s.kslwtseq,e.kslednam,
e.ksledp1,s.kslwtp1,s.kslwtp1r,e.ksledp2,
s.kslwtp2,s.kslwtp2r,e.ksledp3,s.kslwtp3,s.kslwtp3r, e.ksledclassid,
e.ksledclass#, e.ksledclass, decode(s.kslwtinwait,
0,decode(bitand(s.kslwtflags,256), 0,-2,
decode(round(s.kslwtstime/10000), 0,-1,
round(s.kslwtstime/10000))), 0),

VIEW_NAME

VIEW_DEFINITION
---
decode(s.kslwtinwait,0,round((s.kslwtstime+s.kslwtltime)/1000000),
round(s.kslwtstime/1000000)), decode(s.kslwtinwait,1,'WAITING',
decode(bitand(s.kslwtflags,256),0,'WAITED UNKNOWN TIME',
decode(round(s.kslwtstime/10000),0,'WAITED SHORT TIME', 'WAITED KNOWN
TIME'))), s.kslwtstime, decode(s.kslwtinwait,0,to_number(null),
decode(bitand(s.kslwtflags,64),64,0,s.kslwttrem)), s.kslwtltime from x$kslwt s,
x$ksled e where s.kslwtevt=e.indx

GV$SESSION_WAIT_CLASS

VIEW_NAME

VIEW_DEFINITION
---
select s.inst_id, s.kslcssid, s.kslcsser, s.kslcsclsid, s.kslcscls,
s.kslcsclsname, s.kslcswts, round(s.kslcstim / 10000) from x$kslcs s where
s.kslcswts != 0

GV$SESSION_WAIT_HISTORY
select s.inst_id,s.kslwhsid,s.kslwhridx,s.kslwhevt,
s.kslwhetext,s.kslwhp1text,s.kslwhp1,s.kslwhp2text,s.kslwhp2,
s.kslwhp3text,s.kslwhp3, round(s.kslwhstime/10000),s.kslwhstime,s.kslwhltime
from x$kslwh s

VIEW_NAME

VIEW_DEFINITION

GV$SESSMETRIC
SELECT inst_id, begtime, endtime, intsize_csec, sessid, sernum, cpu,
phyrds, logrds, pga_memory, hard_parses, soft_parses, phyrds_pct,
logrds_pct FROM x$kewmsemv WHERE flag1 = 1

GV$SESSTAT
select inst_id,ksusenum,ksusestn,ksusestv from x$ksusesta where
bitand(ksspaflg,1)!=0 and bitand(ksuseflg,1)!=0 and ksusestn<(select ksusgstl

VIEW_NAME

VIEW_DEFINITION
---
from x$ksusgif)

GV$SESS_IO
select inst_id,indx, ksusesbg, ksusescg, ksusespr, ksusesbc, ksusescc from
x$ksusio where bitand(ksspaflg,1)!=0 and bitand(ksuseflg,1)!=0

GV$SESS_TIME_MODEL
select map.inst_id, sesv.ksusenum, map.extid, map.sname, sesv.kewsval from
x$kewssmap map, x$kewssesv sesv where map.soffst = sesv.kewsnum and map.aggid =

VIEW_NAME

VIEW_DEFINITION
---
1 and bitand(sesv.ksspaflg,1)!=0 and bitand(sesv.ksuseflg,1)!=0 and (map.stype
= 2 or map.stype = 3)

GV$SES_OPTIMIZER_ENV
select INST_ID, SID_QKSCESEROW, PNUM_QKSCESEROW,
PNAME_QKSCESEROW, FID_QKSCESEROW,
decode(bitand(FLAGS_QKSCESEROW, 2), 0, 'NO', 'YES'), PVALUE_QKSCESEROW
from X$QKSCESES where
bitand(FLAGS_QKSCESEROW, 8) = 0 and

VIEW_NAME

VIEW_DEFINITION

(bitand(FLAGS_QKSCESEROW, 4) = 0 or
bitand(FLAGS_QKSCESEROW, 2) = 0)

GV$SGA
select inst_id,ksmsdnam,ksmsdval from x$ksmsd

GV$SGAINFO
select b, c, d, e from (select ksmsgmemidx a, inst_id b, ksmsgmemnam c,
ksmsgmemval d, decode(ksmsgmemrez, 0, 'No', 1, 'Yes', NULL) e from

VIEW_NAME

VIEW_DEFINITION
---
x$ksmsgmem union select 32 a, USERENV('Instance') b, 'Free SGA Memory

```
Available' c,    current_size d, NULL e from v$sga_dynamic_free_memory)

GV$SGASTAT
select inst_id,'',ksmssnam,ksmsslen from x$ksmfs where ksmsslen>1  union all
select inst_id,'shared pool',ksmssnam, sum(ksmsslen) from x$ksmss    where
ksmsslen>1 group by inst_id, 'shared pool', ksmssnam  union all  select
inst_id,'large pool',ksmssnam, sum(ksmsslen) from x$ksmls    where ksmsslen>1
group by inst_id, 'large pool', ksmssnam  union all  select inst_id,'java

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
pool',ksmssnam, sum(ksmsslen) from x$ksmjs    where ksmsslen>1 group by inst_id,
'java pool', ksmssnam  union all  select inst_id,'streams pool',ksmssnam,
sum(ksmsslen) from x$ksmstrs    where ksmsslen>1 group by inst_id, 'streams
pool', ksmssnam

GV$SGA_CURRENT_RESIZE_OPS
select sc.inst_id, sc.component,    decode(sc.opcode, 0, 'STATIC', 1,
'INITIALIZING', 2, 'DISABLED',          3, 'GROW', 4, 'SHRINK', 5,
'SHRINK_CANCEL', NULL),    decode(sc.opmode, 1, 'MANUAL', 2, 'DEFERRED', 3,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
'IMMEDIATE', NULL),    pn.name, sc.initsize * sc.gransize, sc.targsize *
sc.gransize,    sc.cursize * sc.gransize, sc.starttime, sc.lasttime   from
x$kmgsct sc, v$parameter pn   where (sc.parno = pn.num)    and (sc.opcode <> 0)
and (sc.starttime is not null)    and (sc.component != 'SGA Target')    and
(sc.component != 'PGA Target')

GV$SGA_DYNAMIC_COMPONENTS
select st.inst_id, st.component, st.cursize * st.gransize,    st.minsize *
st.gransize, st.maxsize * st.gransize,    st.usersize * st.gransize,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
st.opercnt,    decode(st.lastoper, 0, 'STATIC', 1, 'INITIALIZING', 2,
'DISABLED',          3, 'GROW', 4, 'SHRINK', 5, 'SHRINK_CANCEL', NULL),
decode(st.lastmode, 1, 'MANUAL', 2, 'DEFERRED', 3, 'IMMEDIATE', NULL),
st.lasttime, st.gransize    from x$kmgsct st where    (st.component != 'SGA
Target') and    (st.component != 'PGA Target')

GV$SGA_DYNAMIC_FREE_MEMORY
select inst_id, gv.gransize * (select count(*) from x$ksmge    where granstate
= 'FREE' or granstate = 'INVALID')   from x$kmgsct gv where rownum=1

VIEW_NAME
```

VIEW_DEFINITION

GV$SGA_RESIZE_OPS
select op.inst_id, gv.component, decode(op.opcode, 0, 'STATIC', 1,
'INITIALIZING', 2, 'DISABLED', 3, 'GROW', 4, 'SHRINK', 5,
'SHRINK_CANCEL', NULL), decode(op.opmode, 1, 'MANUAL', 2, 'DEFERRED', 3,
'IMMEDIATE', NULL), pn.name, op.initsize * gv.gransize, op.targsize *
gv.gransize, op.realsize * gv.gransize, decode(op.status, 0,
'INACTIVE', 1, 'PENDING', 2, 'COMPLETE', 3, 'CANCELLED', 4, 'ERROR',
5, 'ERROR', 6, 'CANCELLED', 7, 'CANCELLED', NULL), op.starttime,

VIEW_NAME

VIEW_DEFINITION
---
op.endtime from x$kmgsop op, x$kmgsct gv, v$parameter pn where (op.grantype
= gv.grantype) and (op.parno = pn.num) and (gv.component != 'SGA
Target') and (gv.component != 'PGA Target') order by op.starttime

GV$SGA_TARGET_ADVICE
select A.inst_id, A.sgasz,
round((A.sgasz / A.base_sgasz), 4),
decode(A.base_estd_dbtime,0,to_number(null),
round(A.base_dbtime * round((A.dbtime / A.base_estd_dbtime), 4), 0)),

VIEW_NAME

VIEW_DEFINITION
---
decode(A.base_estd_dbtime,0,to_number(null),
round((A.dbtime / A.base_estd_dbtime), 4)),
decode(A.base_estd_phy_reads,0,to_number(null),
round(A.base_phy_reads *
round((A.estd_physical_reads / A.base_estd_phy_reads), 4), 0)) from
x$kmgsbsadv A order
by A.inst_id

GV$SHARED_POOL_ADVICE

VIEW_NAME

VIEW_DEFINITION
---
select inst_id, sp_size, round(sp_size / basesp_size, 4), kglsim_size,
kglsim_objs, kglsim_timesave, decode(kglsim_basetimesave, 0, to_number(null),
round(kglsim_timesave / kglsim_basetimesave, 4)), kglsim_parsetime,
decode(kglsim_baseparsetime, 0, to_number(null), round(kglsim_parsetime
/ kglsim_baseparsetime, 4)), kglsim_hits from x$kglsim

GV$SHARED_POOL_RESERVED

select p.inst_id, p.free_space, p.avg_free_size, p.free_count, p.max_free_size,
p.used_size, p.avg_used_size, p.used_count, p.max_used_size, s.requests,

VIEW_NAME

VIEW_DEFINITION
----
s.request_misses, s.last_miss_size, s.max_miss_size, s.request_failures,
s.last_failure_size, s.aborted_request_threshold, s.aborted_requests,
s.last_aborted_size from (select avg(x$ksmspr.inst_id) inst_id,
sum(decode(ksmchcls,'R-free',ksmchsiz,0)) free_space,
avg(decode(ksmchcls,'R-free',ksmchsiz,0)) avg_free_size,
sum(decode(ksmchcls,'R-free',1,0)) free_count,
max(decode(ksmchcls,'R-free',ksmchsiz,0)) max_free_size,
sum(decode(ksmchcls,'R-free',0,ksmchsiz)) used_size,
avg(decode(ksmchcls,'R-free',0,ksmchsiz)) avg_used_size,

VIEW_NAME

VIEW_DEFINITION
----
sum(decode(ksmchcls,'R-free',0,1)) used_count,
max(decode(ksmchcls,'R-free',0,ksmchsiz)) max_used_size from x$ksmspr where
ksmchcom not like '%reserved sto%') p, (select sum(kghlurcn) requests,
sum(kghlurmi) request_misses, max(kghlurmz) last_miss_size, max(kghlurmx)
max_miss_size, sum(kghlunfu) request_failures, max(kghlunfs) last_failure_size,
max(kghlumxa) aborted_request_threshold, sum(kghlumer) aborted_requests,
max(kghlumes) last_aborted_size from x$kghlu) s

GV$SHARED_SERVER

VIEW_NAME

VIEW_DEFINITION
----
select inst_id,kmmsinam,kmmsiprp,kmmsista,kmmsinmg,
kmmsinmb,kmmsibrk,kmmsivcp,kmmsiidl,kmmsibsy,kmmsitnc from x$kmmsi where
bitand(kmmsiflg,1)!=0

GV$SHARED_SERVER_MONITOR
select inst_id,kmmsgcmx,kmmsgmmx,kmmsgsta,kmmsgtrm,kmmsgsmx from x$kmmsg

GV$SORT_SEGMENT
select inst_id, tablespace_name, segment_file, segment_block, extent_size,

VIEW_NAME

VIEW_DEFINITION
----
current_users, total_extents, total_blocks, used_extents, used_blocks,
free_extents, free_blocks, added_extents, extent_hits, freed_extents,
free_requests, max_size, max_blocks, max_used_size, max_used_blocks,

max_sort_size, max_sort_blocks, relative_fno from x$ktstssd

GV$SORT_USAGE
select x$ktsso.inst_id, username, username, ktssoses, ktssosno, prev_sql_addr,
prev_hash_value, prev_sql_id, ktssotsn, decode(ktssocnt, 0, 'PERMANENT', 1,
'TEMPORARY'), decode(ktssosegt, 1, 'SORT', 2, 'HASH', 3, 'DATA', 4, 'INDEX', 5,

VIEW_NAME

VIEW_DEFINITION

'LOB_DATA', 6, 'LOB_INDEX' , 'UNDEFINED'), ktssofno, ktssobno, ktssoexts,
ktssoblks, ktssorfno from x$ktsso, v$session where ktssoses = v$session.saddr
and ktssosno = v$session.serial#

GV$SPPARAMETER
select INST_ID, KSPSPFFTCTXSPSID, KSPSPFFTCTXSPNAME,
decode(KSPSPFFTCTXPARTYP,1,'boolean',2,'string', 3,'integer',
4,'file',5,'number', 6,'big integer', 'unknown'), KSPSPFFTCTXSPVALUE,
KSPSPFFTCTXSPDVALUE, KSPSPFFTCTXISSPECIFIED, KSPSPFFTCTXORDINAL,

VIEW_NAME

VIEW_DEFINITION

KSPSPFFTCTXCOMMENT from x$kspspfile WHERE
((translate(KSPSPFFTCTXSPNAME,'_','#') not like '##%') and
((translate(KSPSPFFTCTXSPNAME, '_', '#') not like '#%') OR
KSPSPFFTCTXISSPECIFIED = 'TRUE'))

GV$SQL
select inst_id,kglnaobj,kglfnobj,kglobt03,
kglobhs0+kglobhs1+kglobhs2+kglobhs3+kglobhs4+kglobhs5+kglobhs6+kglobt16,
kglobt08+kglobt11, kglobt10, kglobt01, decode(kglobhs6,0,0,1),

VIEW_NAME

VIEW_DEFINITION
---
decode(kglhdlmd,0,0,1), kglhdlkc, kglobt04, kglobt05, kglobt48, kglobt35,
kglobpc6, kglhdldc, substr(to_char(kglnatim,'YYYY-MM-DD/HH24:MI:SS'),1,19),
kglhdivc, kglobt12, kglobt13, kglobwdw, kglobt14, kglobwap, kglobwcc, kglobwcl,
kglobwui, kglobt42, kglobt43, kglobt15, kglobt02, decode(kglobt32, 0,
'NONE', 1, 'ALL_ROWS', 2, 'FIRST_ROWS', 3, 'RULE',
4, 'CHOOSE', 'UNKNOWN'), kglobtn0, kglobcce, kglobcceh, kglobt17,
kglobt18, kglobts4, kglhdkmk, kglhdpar, kglobtp0, kglnahsh, kglobt46, kglobt30,
kglobt09, kglobts5, kglobt48, kglobts0, kglobt19, kglobts1, kglobt20, kglobt21,
kglobts2, kglobt06, kglobt07, decode(kglobt28, 0, to_number(NULL), kglobt28),

VIEW_NAME

VIEW_DEFINITION

```
--------------------------------------------------------------------------------
kglhdadr, kglobt29, decode(bitand(kglobt00,64),64, 'Y', 'N'), decode(kglobsta,
1, 'VALID',        2, 'VALID_AUTH_ERROR',        3, 'VALID_COMPILE_ERROR',
4, 'VALID_UNAUTH',        5, 'INVALID_UNAUTH',        6, 'INVALID'), kglobt31,
substr(to_char(kglobtt0,'YYYY-MM-DD/HH24:MI:SS'),1,19), decode(kglobt33, 1, 'Y',
'N'),  decode(bitand(kglobacs, 1), 1, 'Y', 'N'),  decode(bitand(kglobacs, 2), 2,
'Y', 'N'),  decode(bitand(kglobacs, 4), 4, 'Y', 'N'),  kglhdclt, kglobts3,
kglobts7, kglobts6, kglobt44, kglobt45,  kglobt47, kglobt49, kglobcla,
kglobcbca, kglobt22  from x$kglcursor_child


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$SQLAREA
select inst_id, kglnaobj, kglfnobj, kglobt03,
kglobhs0+kglobhs1+kglobhs2+kglobhs3+kglobhs4+kglobhs5+kglobhs6,
kglobt08+kglobt11, kglobt10, kglobt01, kglobccc, kglobclc, kglhdlmd, kglhdlkc,
kglobt04, kglobt05, kglobt48, kglobt35, kglobpc6, kglhdldc,
substr(to_char(kglnatim,'YYYY-MM-DD/HH24:MI:SS'),1,19), kglhdivc, kglobt12,
kglobt13, kglobwdw, kglobt14, kglobwap, kglobwcc, kglobwcl, kglobwui, kglobt42,
kglobt43, kglobt15, kglobt02, decode(kglobt32, 0, 'NONE',                1,
'ALL_ROWS',                2, 'FIRST_ROWS',                3, 'RULE',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
4, 'CHOOSE', 'UNKNOWN'), kglobtn0, kglobcce, kglobcceh, kglobt17, kglobt18,
kglobts4, kglhdkmk, kglhdpar, kglnahsh, kglobt46, kglobt30, kglobts0, kglobt19,
kglobts1, kglobt20, kglobt21, kglobts2, kglobt06, kglobt07, decode(kglobt28, 0,
NULL, kglobt28), kglhdadr, decode(bitand(kglobt00,64),64, 'Y', 'N'),
decode(kglobsta,        1, 'VALID',        2, 'VALID_AUTH_ERROR',        3,
'VALID_COMPILE_ERROR',        4, 'VALID_UNAUTH',        5, 'INVALID_UNAUTH',
6, 'INVALID'), kglobt31, kglobtt0, decode(kglobt33, 1, 'Y', 'N'),
decode(bitand(kglobacs, 1), 1, 'Y', 'N'),  decode(bitand(kglobacs, 2), 2, 'Y',
'N'),  decode(bitand(kglobacs, 4), 4, 'Y', 'N'),  kglhdclt, kglobts3, kglobts7,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
kglobts6, kglobt44, kglobt45, kglobt47, kglobt49, kglobcla, kglobcbca, kglobt22
from  x$kglcursor_child_sqlid  where kglobt02 != 0

GV$SQLAREA_PLAN_HASH
select inst_id, kglnaobj, kglfnobj, kglhdpar, kglobt46, kglobt03, kglobt30,
kglobccc, kglhdadr,
kglobhs0+kglobhs1+kglobhs2+kglobhs3+kglobhs4+kglobhs5+kglobhs6,
kglobt08+kglobt11, kglobt10, kglobt01, kglobclc, kglhdlmd, kglhdlkc, kglobpc6,
kglobt04, kglobt05, kglobt50, kglobt35, kglhdldc, kglnatim, kglobtt0, kglobcla,
```

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
kglhdivc, kglobt12, kglobt13, kglobwdw, kglobt14, kglobt06, kglobt07, kglobwap,
kglobwcc, kglobwcl, kglobwui, kglobt42, kglobt43, kglobt15, kglobt02,
decode(kglobt32, 0, 'NONE',               1, 'ALL_ROWS',               2,
'FIRST_ROWS',               3, 'RULE',               4, 'CHOOSE',
'UNKNOWN'), kglobtn0, kglobcce, kglobcceh, kglobt17, kglobt18, kglobts4,
kglhdkmk, kglobts0, kglobt19, kglobts1, kglobt20, kglobt21, kglobts2,
decode(kglobt28, 0, NULL, kglobt28), decode(bitand(kglobt00,64),64, 'Y', 'N'),
decode(kglobsta,       1, 'VALID',       2, 'VALID_AUTH_ERROR',       3,
'VALID_COMPILE_ERROR',       4, 'VALID_UNAUTH',       5, 'INVALID_UNAUTH',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
6, 'INVALID'), kglobt31, kglobts3, kglobt44, kglobt45, kglobt47, kglobt49,
kglobcbca, kglobt22  from  x$kglcursor_child_sqlidph

GV$SQLFN_ARG_METADATA
select inst_id,       id,       argnum,       case when dtype = 0 then
'UNKNOWN'       when dtype = 1 then 'NUMERIC'       when dtype = 2 then
'STRING'       when dtype = 3 then 'DATETYPE'       when dtype = 4 then
'BINARY'       when dtype = 5 then 'EXPR'       when dtype = 6 then 'ARG 1'
when dtype = 7 then 'ARG 2'       when dtype = 8 then 'ARG 3'       else

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
'INVALID'       end,       descr       from  x$oparg

GV$SQLFN_METADATA
select o.inst_id,       o.id,       o.name,       o.operands,
o.maxoperands,       case when d.ret_type = 0 then 'UNKNOWN'       when
d.ret_type = 1 then 'NUMERIC'       when d.ret_type = 2 then 'STRING'
when d.ret_type = 3 then 'DATETYPE'       when d.ret_type = 4 then 'BINARY'
when d.ret_type = 5 then 'EXPR'       when d.ret_type = 6 then 'ARG 1'
when d.ret_type = 7 then 'ARG 2'       when d.ret_type = 8 then 'ARG 3'

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
else 'INVALID'       end,       case when v.version = 1 then 'V6 Oracle'
when v.version = 2 then 'SQL/DS'       when v.version = 10 then 'V71 Oracle'
when v.version = 11 then 'V73 Oracle'       when v.version = 12 then 'V80
Oracle'       when v.version = 13 then 'V81 Oracle'       when v.version =
14 then 'V816 Oracle'       when v.version = 16 then 'V82 Oracle'       when
```

v.version = 18 then 'V92 Oracle' when v.version = 19 then 'V10 Oracle'
when v.version = 20 then 'V10R2 Oracle' when v.version = 21 then 'V11R1
Oracle' else 'INVALID' end, case when bitand(o.flags, 4096)
= 4096 then 'YES' when bitand(o.flags, 32768) = 32768 then 'YES'

VIEW_NAME

VIEW_DEFINITION
---
else 'NO' end, case when bitand(o.flags, 8) = 8 then 'YES'
else 'NO' end, case when d.disp_type = 0 then 'UNKNOWN'
when d.disp_type = 1 then 'NORMAL' when d.disp_type = 2 then
'ARITHMATIC' when d.disp_type = 3 then 'PARENTHESIS' when
d.disp_type = 4 then 'REL-OP' when d.disp_type = 5 then 'CASELIKE'
when d.disp_type = 6 then 'NOPARENTHESIS' else 'INVALID' end,
d.usg, d.descr from x$operators o, x$opversion v, x$opdesc d
where o.indx = v.indx and v.indx = d.indx and o.inst_id =
v.inst_id and v.inst_id = d.inst_id

VIEW_NAME

VIEW_DEFINITION

GV$SQLSTATS
select INST_ID, SQL_TEXT, SQL_FULLTEXT, SQL_ID, LAST_ACTIVE_TIME,
LAST_ACTIVE_CHILD_ADDRESS, PLAN_HASH_VALUE, PARSE_CALLS, DISK_READS,
DIRECT_WRITES, BUFFER_GETS, ROWS_PROCESSED, SERIALIZABLE_ABORTS, FETCHES,
EXECUTIONS, END_OF_FETCH_COUNT, LOADS, VERSION_COUNT, INVALIDATIONS,
PX_SERVERS_EXECUTIONS, CPU_TIME, ELAPSED_TIME, AVG_HARD_PARSE_TIME,
APPLICATION_WAIT_TIME, CONCURRENCY_WAIT_TIME, CLUSTER_WAIT_TIME,
USER_IO_WAIT_TIME, PLSQL_EXEC_TIME, JAVA_EXEC_TIME, SORTS, SHARABLE_MEM,

VIEW_NAME

VIEW_DEFINITION

TOTAL_SHARABLE_MEM, TYPECHECK_MEM FROM x$kkssqlstat

GV$SQLTEXT
select inst_id,kglhdadr, kglnahsh, kglnasqlid, kgloboct, piece, name from
x$kglna where kgloboct != 0

GV$SQLTEXT_WITH_NEWLINES
select inst_id,kglhdadr, kglnahsh, kglnasqlid, kgloboct, piece, name from
x$kglna1 where kgloboct != 0

VIEW_NAME

VIEW_DEFINITION

GV$SQL_BIND_DATA
select inst_id,kxsbdcur, kxsbdbnd, kxsbddty, kxsbdmxl, kxsbdpmx, kxsbdmal,
kxsbdpre, kxsbdscl, kxsbdofl, kxsbdof2, kxsbdbfp, kxsbdbln, kxsbdavl, kxsbdbfl,
kxsbdind, kxsbdval from x$kxsbd

GV$SQL_BIND_METADATA
select inst_id,kglhdadr, position, kkscbndt, kkscbndl, kkscbnda, kksbvnnam from
x$kksbv

VIEW_NAME

VIEW_DEFINITION

GV$SQL_CS_HISTOGRAM
select inst_id, phadd_kkocs, hashv_kkocs, sqlid_kkocs,
childno_kkocs, bucketid_kkocs, count_kkocs from X$KKOCS_HISTOGRAM

GV$SQL_CS_SELECTIVITY
select inst_id, phadd_kkocs, hashv_kkocs, sqlid_kkocs,
childno_kkocs, pred_kkocs, rangeid_kkocs, lowsel_kkocs, highsel_kkocs
from X$KKOCS_SELECTIVITY

VIEW_NAME

VIEW_DEFINITION

GV$SQL_CS_STATISTICS
select inst_id, phadd_kkocs, hashv_kkocs, sqlid_kkocs,
childno_kkocs, bsethv_kkocs, peeked_kkocs, execount_kkocs,
rowsproc_kkocs, buffgets_kkocs, cputime_kkocs from X$KKOCS_STATISTICS

GV$SQL_CURSOR
select inst_id,kxscccur, kxscccfl, decode(kxsccsta, 0, 'CURNULL', 1,
'CURSYNTAX', 2, 'CURPARSE', 3, 'CURBOUND', 4, 'CURFETCH', 5, 'CURROW', 'ERROR'),

VIEW_NAME

VIEW_DEFINITION
---
kxsccphd, kxsccplk, kxsccclk, kxscccpn, kxscctbm, kxscctwm, kxscctbv, kxscctdv,
kxsccbdf, kxsccflg, kxsccfl2, kxsccchd from x$kxscc

GV$SQL_FEATURE
select INST_ID, ID_QKSFMSYROW, DESC_QKSFMSYROW,
PROPS_QKSFMSYROW from x$qksfm

GV$SQL_FEATURE_DEPENDENCY
select INST_ID, ID_QKSFMDEPSYROW, DID_QKSFMDEPSYROW from

VIEW_NAME

```
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
x$qksfmdep

GV$SQL_FEATURE_HIERARCHY
select  INST_ID,       ID_QKSFMPRTSYROW,       PID_QKSFMPRTSYROW from
x$qksfmprt

GV$SQL_HINT
select  INST_ID,       TOKEN_QKSHTSYROW,       FID_QKSHTSYROW,
CLASS_QKSHTSYROW,       INVERSE_QKSHTSYROW,       LEVEL_QKSHTSYROW,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
PROPS_QKSHTSYROW,       VERSION_QKSHTSYROW,       VERSION_OL_QKSHTSYROW from
x$qksht

GV$SQL_JOIN_FILTER
SELECT  INST_ID, QCSID, QCINSTID, SQLHASHV,               LEN, NSET, FLT,
TOT, ACTIVE         FROM X$QESBLSTAT WHERE QCINSTID != 0

GV$SQL_MONITOR
select  inst_id,       key_keswxmon,       case status_keswxmon when 1 then

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
'EXECUTING'                       when 2 then 'DONE (ERROR)'
when 3 then 'DONE (FIRST N ROWS)'                       when 4 then 'DONE
(ALL ROWS)'                       when 5 then 'DONE'
when 6 then 'FREED'                       else 'UNKNOWN' end,
startmon_keswxmon,       lastpub_keswxmon,       pubcount_keswxmon,
sid_keswxmon,       procname_keswxmon,       sqlid_keswxmon,
execstart_keswxmon,       execid_keswxmon,       planhash_keswxmon,
childaddr_keswxmon,       serial_keswxmon,       decode(pxsrvnum_keswxmon,
65535, to_number(NULL),             pxsrvnum_keswxmon),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(pxsrvnum_keswxmon, 65535, to_number(NULL),
pxsrvgrp_keswxmon),       decode(pxsrvnum_keswxmon, 65535, to_number(NULL),
pxsrvset_keswxmon),       decode(pxsrvnum_keswxmon, 65535, to_number(NULL),
pxqcinstid_keswxmon),       decode(pxsrvnum_keswxmon, 65535, to_number(NULL),
pxqcsessid_keswxmon),       elapsed_time,       cpu_time,       fetches,
buffer_gets,       disk_reads,       direct_writes,
application_wait_time,       concurrency_wait_time,       cluster_wait_time,
```

user_io_wait_time, plsql_exec_time, java_exec_time from
X$KESWXMON

VIEW_NAME

VIEW_DEFINITION

GV$SQL_OPTIMIZER_ENV
select INST_ID, KQLFSQCE_PHAD, KQLFSQCE_HASH,
KQLFSQCE_SQLID, KQLFSQCE_HADD, KQLFSQCE_CHNO,
KQLFSQCE_PNUM, KQLFSQCE_PNAME,
decode(bitand(KQLFSQCE_FLAGS, 2), 0, 'NO', 'YES'), KQLFSQCE_PVALUE
from X$KQLFSQCE where
bitand(KQLFSQCE_FLAGS, 8) = 0 and
(bitand(KQLFSQCE_FLAGS, 4) = 0 or

VIEW_NAME

VIEW_DEFINITION
----
bitand(KQLFSQCE_FLAGS, 2) = 0)

GV$SQL_PLAN
select inst_id, kqlfxpl_phad, kqlfxpl_hash,
kqlfxpl_sqlid, kqlfxpl_plhash, kqlfxpl_hadd,
kqlfxpl_chno, kqlfxpl_timestamp, substr(kqlfxpl_oper, 1, 30),
substr(kqlfxpl_oopt, 1, 30), substr(kqlfxpl_tqid, 1, 40),
to_number(decode(kqlfxpl_objn, 0, NULL, kqlfxpl_objn)),
kqlfxpl_objowner, kqlfxpl_objname,

VIEW_NAME

VIEW_DEFINITION
----
kqlfxpl_alias, substr(kqlfxpl_objtype, 1, 20),
substr(kqlfxpl_opti, 1, 20), kqlfxpl_opid,
to_number(decode(kqlfxpl_opid, 0, NULL, kqlfxpl_paid)), kqlfxpl_depth,
to_number(decode(kqlfxpl_pos, 0, decode(kqlfxpl_cost,
4294967295, NULL, kqlfxpl_cost),
kqlfxpl_pos)), kqlfxpl_scols, to_number(decode(kqlfxpl_cost,
4294967295, NULL, kqlfxpl_cost)),
to_number(decode(kqlfxpl_card, 0, NULL, kqlfxpl_card)),
to_number(decode(kqlfxpl_size, 0, NULL, kqlfxpl_size)),

VIEW_NAME

VIEW_DEFINITION
----
substr(kqlfxpl_otag, 1, 35), substr(kqlfxpl_psta, 1, 64),
substr(kqlfxpl_psto, 1, 64), to_number(decode(kqlfxpl_pnid, 0, NULL,
kqlfxpl_pnid)), kqlfxpl_other, substr(kqlfxpl_dist, 1, 20),

```
to_number(decode(kqlfxpl_cpuc, 4294967295, NULL,
kqlfxpl_cpuc)),
to_number(decode(kqlfxpl_ioct, 4294967295, NULL,
kqlfxpl_ioct)),         to_number(decode(kqlfxpl_temp, 0, NULL, kqlfxpl_temp)),
kqlfxpl_keys,           kqlfxpl_filter,         kqlfxpl_proj,
to_number(decode(kqlfxpl_time, 0, NULL, kqlfxpl_time)),         kqlfxpl_qblock,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
kqlfxpl_remark, kqlfxpl_other_xml         from x$kqlfxpl p

GV$SQL_PLAN_MONITOR
select  inst_id,        key_keswxmonp,        case status_keswxmonp when 1
then 'EXECUTING'                          when 2 then 'DONE (ERROR)'
when 3 then 'DONE (FIRST N ROWS)'                               when 4 then
'DONE (ALL ROWS)'                         when 5 then 'DONE'
when 6 then 'FREED'                        else 'UNKNOWN' end,
startmon_keswxmonp,        lastpub_keswxmonp,        firstchg_keswxmonp,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
lastchg_keswxmonp,        pubcount_keswxmonp,        sid_keswxmonp,
procname_keswxmonp,        sqlid_keswxmonp,        execstart_keswxmonp,
execid_keswxmonp,        planhash_keswxmonp,        childaddr_keswxmonp,
lineid_keswxmonp,        lineopnam_keswxmonp,        lineopopt_keswxmonp,
nsta_keswxmonp,        nrows_keswxmonp,        case wasta_keswxmonp when 1
then mem_keswxmonp * 1024                         else null end,
case wasta_keswxmonp when 0 then null                               else
maxmem_keswxmonp * 1024 end,        case when wasta_keswxmonp = 1 and
tmp_keswxmonp != 0            then tmp_keswxmonp * 1024 else null end,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
case when wasta_keswxmonp != 0 and maxtmp_keswxmonp != 0            then
maxtmp_keswxmonp * 1024 else null end from X$KESWXMON_PLAN

GV$SQL_PLAN_STATISTICS
select inst_id,          PHADD_QESRS,             HASHV_QESRS,
SQLID_QESRS,             PLHASH_QESRS,            HADDR_QESRS,
CHILDNO_QESRS,           OPERID_QESRS,            EXECS_QESRS,
LSTARTS_QESRS,           STARTS_QESRS,            LOUTROWS_QESRS,
OUTROWS_QESRS,           LCRGETS_QESRS,           CRGETS_QESRS,

VIEW_NAME
------------------------------
VIEW_DEFINITION
```

```
--------------------------------------------------------------------------------
LCUGETS_QESRS,          CUGETS_QESRS,          LDREADS_QESRS,
DREADS_QESRS,           LDWRITES_QESRS,        DWRITES_QESRS,
LELAPTIME_QESRS,        ELAPTIME_QESRS         from X$QESRSTAT

GV$SQL_PLAN_STATISTICS_ALL
select inst_id, PHADD_QESRS, HASHV_QESRS, SQLID_QESRS, PLHASH_QESRS,
HADDR_QESRS, CHILDNO_QESRS, TIMESTAMP_QESRS, substr(oper_qesrs, 1, 30),
substr(oopt_qesrs, 1, 30), substr(tqid_qesrs, 1, 40),
to_number(decode(objn_qesrs, 0, NULL, objn_qesrs)), objowner_qesrs,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
objname_qesrs, alias_qesrs, substr(objtype_qesrs, 1, 20), substr(opti_qesrs,
1, 20), opid_qesrs, to_number(decode(opid_qesrs, 0, NULL, paid_qesrs)),
depth_qesrs, to_number(decode(pos_qesrs, 0, decode(cost_qesrs, 4294967295,
NULL,     cost_qesrs), pos_qesrs)), scols_qesrs, to_number(decode(cost_qesrs,
4294967295, NULL, cost_qesrs)), to_number(decode(card_qesrs, 0, NULL,
card_qesrs)), to_number(decode(size_qesrs, 0, NULL, size_qesrs)),
substr(otag_qesrs, 1, 35), substr(psta_qesrs, 1, 64), substr(psto_qesrs, 1,
64), to_number(decode(pnid_qesrs, 0, NULL, pnid_qesrs)), other_qesrs,
substr(dist_qesrs, 1, 20),    to_number(decode(cpuc_qesrs, 4294967295, NULL,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
cpuc_qesrs)), to_number(decode(ioct_qesrs, 4294967295, NULL, ioct_qesrs)),
to_number(decode(temp_qesrs, 0, NULL, temp_qesrs)), KEYS_QESRS, FILTER_QESRS,
PROJ_QESRS, to_number(decode(time_qesrs, 0, NULL, time_qesrs)), QBLOCK_QESRS,
REMARK_QESRS,  OTHER_XML_QESRS,  EXECS_QESRS,
to_number(decode(LSTARTS_QESRS, 4294967295, NULL, LSTARTS_QESRS)),
to_number(decode(LSTARTS_QESRS, 4294967295, NULL, STARTS_QESRS)),
to_number(decode(LSTARTS_QESRS, 4294967295, NULL, LOUTROWS_QESRS)),
to_number(decode(LSTARTS_QESRS, 4294967295, NULL, OUTROWS_QESRS)),
to_number(decode(LSTARTS_QESRS, 4294967295, NULL, LCRGETS_QESRS)),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
to_number(decode(LSTARTS_QESRS, 4294967295, NULL, CRGETS_QESRS)),
to_number(decode(LSTARTS_QESRS, 4294967295, NULL, LCUGETS_QESRS)),
to_number(decode(LSTARTS_QESRS, 4294967295, NULL, CUGETS_QESRS)),
to_number(decode(LSTARTS_QESRS, 4294967295, NULL, LDREADS_QESRS)),
to_number(decode(LSTARTS_QESRS, 4294967295, NULL, DREADS_QESRS)),
to_number(decode(LSTARTS_QESRS, 4294967295, NULL, LDWRITES_QESRS)),
to_number(decode(LSTARTS_QESRS, 4294967295, NULL, DWRITES_QESRS)),
to_number(decode(LSTARTS_QESRS, 4294967295, NULL, LELAPTIME_QESRS)),
to_number(decode(LSTARTS_QESRS, 4294967295, NULL, ELAPTIME_QESRS)),
```

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
substr(SIZEPOLICY_QESRS, 1, 10),  to_number(decode(OPTIMAL_QESRS, 0, NULL,
OPTIMAL_QESRS * 1024)),  to_number(decode(OPTIMAL_QESRS, 0, NULL, ONEPASS_QESRS
* 1024)),  to_number(decode(OPTIMAL_QESRS, 0, NULL, LASTMEM_QESRS * 1024)),
decode(OPTIMAL_QESRS, 0, NULL,    substr(decode(LASTPASS_QESRS, 0, 'OPTIMAL',
to_char(LASTPASS_QESRS) || ' PASS' ||      decode(LASTPASS_QESRS, 1, '', 'ES')),
1, 10)),  to_number(decode(LASTDOP_QESRS, 0, NULL, LASTDOP_QESRS)),
to_number(decode(OPTIMAL_QESRS, 0, NULL, (OPTACTS_QESRS +
SPAACTS_QESRS + MPAACTS_QESRS))),  to_number(decode(OPTIMAL_QESRS, 0, NULL,
OPTACTS_QESRS)),  to_number(decode(OPTIMAL_QESRS, 0, NULL, SPAACTS_QESRS)),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
to_number(decode(OPTIMAL_QESRS, 0, NULL, MPAACTS_QESRS)),
to_number(decode(OPTIMAL_QESRS, 0, NULL, ATIME_QESRS)),
to_number(decode(MAXTSEG_QESRS, 0, NULL, MAXTSEG_QESRS)),
to_number(decode(LASTTSEG_QESRS, 0, NULL, LASTTSEG_QESRS))  from X$QESRSTATALL p
where p.haddr_qesrs != p.phadd_qesrs

GV$SQL_REDIRECTION
select c.inst_id,c.kglhdadr,c.kglhdpar,c.kglnahsh,c.kglobt03,       c.kglobt09,
c.kglobt17,c.kglobt18, c.kglobt02,  decode(r.reason,1,'INVALID OBJECT',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
2,'ROWID',3,'QUERY REWRITE','READ ONLY'), r.error_code, r.position,
r.sql_text_piece, r.error_msg  from x$kglcursor_child c, x$kkssrd r  where
c.kglhdpar = r.parAddr and c.kglhdadr = r.kglhdadr

GV$SQL_SHARED_CURSOR
select inst_id, sql_id, kglhdpar, kglhdadr, childno, decode(bitand(bitvector,
POWER(2,0)), POWER(2,  0), 'Y','N'),decode(bitand(bitvector, POWER(2,1)),
POWER(2,  1), 'Y','N'),decode(bitand(bitvector, POWER(2,2)), POWER(2,  2),
'Y','N'),decode(bitand(bitvector, POWER(2,3)), POWER(2,  3),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
'Y','N'),decode(bitand(bitvector, POWER(2,4)), POWER(2,  4),
'Y','N'),decode(bitand(bitvector, POWER(2,5)), POWER(2,  5),
'Y','N'),decode(bitand(bitvector, POWER(2,6)), POWER(2,  6),
'Y','N'),decode(bitand(bitvector, POWER(2,7)), POWER(2,  7),
'Y','N'),decode(bitand(bitvector, POWER(2,8)), POWER(2,  8),
```

'Y','N'),decode(bitand(bitvector, POWER(2,9)), POWER(2, 9),
'Y','N'),decode(bitand(bitvector, POWER(2,10)), POWER(2, 10),
'Y','N'),decode(bitand(bitvector, POWER(2,11)), POWER(2, 11),
'Y','N'),decode(bitand(bitvector, POWER(2,12)), POWER(2, 12),

VIEW_NAME

VIEW_DEFINITION

'Y','N'),decode(bitand(bitvector, POWER(2,13)), POWER(2, 13),
'Y','N'),decode(bitand(bitvector, POWER(2,14)), POWER(2, 14),
'Y','N'),decode(bitand(bitvector, POWER(2,15)), POWER(2, 15),
'Y','N'),decode(bitand(bitvector, POWER(2,16)), POWER(2, 16),
'Y','N'),decode(bitand(bitvector, POWER(2,17)), POWER(2, 17),
'Y','N'),decode(bitand(bitvector, POWER(2,18)), POWER(2, 18),
'Y','N'),decode(bitand(bitvector, POWER(2,19)), POWER(2, 19), 'Y','N'),
decode(bitand(bitvector, POWER(2,20)), POWER(2, 20),
'Y','N'),decode(bitand(bitvector, POWER(2,21)), POWER(2, 21),

VIEW_NAME

VIEW_DEFINITION

'Y','N'),decode(bitand(bitvector, POWER(2,22)), POWER(2, 22),
'Y','N'),decode(bitand(bitvector, POWER(2,23)), POWER(2, 23),
'Y','N'),decode(bitand(bitvector, POWER(2,24)), POWER(2, 24),
'Y','N'),decode(bitand(bitvector, POWER(2,25)), POWER(2, 25),
'Y','N'),decode(bitand(bitvector, POWER(2,26)), POWER(2, 26),
'Y','N'),decode(bitand(bitvector, POWER(2,27)), POWER(2, 27),
'Y','N'),decode(bitand(bitvector, POWER(2,28)), POWER(2, 28),
'Y','N'),decode(bitand(bitvector, POWER(2,29)), POWER(2, 29),
'Y','N'),decode(bitand(bitvector, POWER(2,30)), POWER(2, 30),

VIEW_NAME

VIEW_DEFINITION

'Y','N'),decode(bitand(bitvector, POWER(2,31)), POWER(2, 31),
'Y','N'),decode(bitand(bitvector, POWER(2,32)), POWER(2, 32),
'Y','N'),decode(bitand(bitvector, POWER(2,33)), POWER(2, 33),
'Y','N'),decode(bitand(bitvector, POWER(2,34)), POWER(2, 34),
'Y','N'),decode(bitand(bitvector, POWER(2,35)), POWER(2, 35),
'Y','N'),decode(bitand(bitvector, POWER(2,36)), POWER(2, 36),
'Y','N'),decode(bitand(bitvector, POWER(2,37)), POWER(2, 37),
'Y','N'),decode(bitand(bitvector, POWER(2,38)), POWER(2, 38), 'Y','N'),
decode(bitand(bitvector, POWER(2,39)), POWER(2, 39),

VIEW_NAME

VIEW_DEFINITION

'Y','N'),decode(bitand(bitvector, POWER(2,40)), POWER(2, 40),

'Y','N'),decode(bitand(bitvector, POWER(2,41)), POWER(2, 41),
'Y','N'),decode(bitand(bitvector, POWER(2,42)), POWER(2, 42),
'Y','N'),decode(bitand(bitvector, POWER(2,43)), POWER(2, 43),
'Y','N'),decode(bitand(bitvector, POWER(2,44)), POWER(2, 44),
'Y','N'),decode(bitand(bitvector, POWER(2,45)), POWER(2, 45),
'Y','N'),decode(bitand(bitvector, POWER(2,46)), POWER(2, 46),
'Y','N'),decode(bitand(bitvector, POWER(2,47)), POWER(2, 47),
'Y','N'),decode(bitand(bitvector, POWER(2,48)), POWER(2, 48),

VIEW_NAME

VIEW_DEFINITION

'Y','N'),decode(bitand(bitvector, POWER(2,49)), POWER(2, 49),
'Y','N'),decode(bitand(bitvector, POWER(2,50)), POWER(2, 50),
'Y','N'),decode(bitand(bitvector, POWER(2,51)), POWER(2, 51),
'Y','N'),decode(bitand(bitvector, POWER(2,52)), POWER(2, 52),
'Y','N'),decode(bitand(bitvector, POWER(2,53)), POWER(2, 53),
'Y','N'),decode(bitand(bitvector, POWER(2,54)), POWER(2, 54),
'Y','N'),decode(bitand(bitvector, POWER(2,55)), POWER(2, 55),
'Y','N'),decode(bitand(bitvector, POWER(2,56)), POWER(2, 56),
'Y','N'),decode(bitand(bitvector, POWER(2,57)), POWER(2, 57),

VIEW_NAME

VIEW_DEFINITION

'Y','N'),decode(bitand(bitvector, POWER(2,58)), POWER(2, 58),
'Y','N'),decode(bitand(bitvector, POWER(2,59)), POWER(2, 59), 'Y','N')from
x$kkscs

GV$SQL_SHARED_MEMORY
select /*+use_nl(h,c)*/ c.inst_id,kglnaobj,kglfnobj, kglnahsh, kglobt03,
kglobhd6, rtrim(substr(ksmchcom, 1, instr(ksmchcom, ':', 1, 1) - 1)),
ltrim(substr(ksmchcom, -(length(ksmchcom) - (instr(ksmchcom, ':',
1, 1))), (length(ksmchcom) - (instr(ksmchcom, ':', 1, 1)) + 1))),

VIEW_NAME

VIEW_DEFINITION
----
ksmchcom, ksmchptr, ksmchsiz, ksmchcls, ksmchtyp, ksmchpar from x$kglcursor c,
x$ksmhp h where ksmchds = kglobhd6 and kglhdadr != kglhdpar

GV$SQL_WORKAREA
SELECT INST_ID, PHADD_QKSMM, HASHV_QKSMM,
SQLID_QKSMM, CHILDNO_QKSMM, WADDR_QKSMM,
substr(OPERTYPE_QKSMM, 1, 20), to_number(decode(OPERTID_QKSMM, 65535,
NULL, OPERTID_QKSMM)), substr(SIZEPOLICY_QKSMM, 1, 10),
OPTIMAL_QKSMM * 1024, ONEPASS_QKSMM * 1024, LASTMEM_QKSMM *

VIEW_NAME

```
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
1024,         substr(decode(LASTPASS_QKSMM, 0, 'OPTIMAL',
to_char(LASTPASS_QKSMM) || ' PASS' ||
decode(LASTPASS_QKSMM, 1, '', 'ES')),               1, 10),
LASTDOP_QKSMM,          (OPTACTS_QKSMM + SPAACTS_QKSMM + MPAACTS_QKSMM),
OPTACTS_QKSMM,          SPAACTS_QKSMM,          MPAACTS_QKSMM,
ATIME_QKSMM,          to_number(decode(MAXTSEG_QKSMM, 0, NULL,
MAXTSEG_QKSMM*1024)),         to_number(decode(LASTTSEG_QKSMM, 0, NULL,
LASTTSEG_QKSMM*1024))         FROM X$QKSMMWDS


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$SQL_WORKAREA_ACTIVE
select INST_ID,          SQLHASHV,          SQLID,          EXECSTA,
decode(execid, 0, to_number(NULL), execid),          WADDR,
substr(OPER_TYPE, 1, 20),          to_number(decode(OPID, 65535, NULL, OPID)),
substr(decode(bitand(MEM_FLAGS,1), 0, 'MANUAL', 'AUTO'), 1, 6),          SID,
to_number(decode(QCINSTID, 65535, NULL, QCINSTID)),
to_number(decode(QCSID, 65535, NULL, QCSID)),          ATIME,          WA_SIZE
* 1024,          to_number(decode(bitand(MEM_FLAGS,1), 0, NULL,
EXP_SIZE*1024)),          ACTUAL_MEM * 1024,          MAX_MEM * 1024,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
PASSES,          to_number(decode(KTSSOTSN, '', NULL, KTSSOSIZE*1024)),
decode(KTSSOTSN, '', NULL, KTSSOTSN),          to_number(decode(KTSSOTSN, '',
NULL, KTSSORFNO)),          to_number(decode(KTSSOTSN, '', NULL, KTSSOBNO))
from   x$qesmmiwt

GV$SQL_WORKAREA_HISTOGRAM
select INST_ID,          LOWBND * 1024,          (HIBND * 1024)-1,
OPTIMAL,          ONEPASS,          MPASS,          MPASS+ONEPASS+OPTIMAL
from   X$QESMMIWH

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

GV$SSCR_SESSIONS
SELECT inst_id, indx, ksusmser, decode(ksusmcrsta, 0, 'NONE', 1, 'MARKED', 2,
'SUSPENDED', 3, 'CAPINIT', 4, 'CAPTURED',5, 'RESINIT', 6, 'RESTORED',7,
'FAILED'),  decode(ksusmcrmod, 0, 'SESSION', 1, 'GLOBAL'),  decode(ksusmcrcsp,
0, 'NONE', 1, 'MINIMAL', 2, 'TYPICAL', 3, 'FULL'),  decode(ksusmcrncc,
-1,'UNKNOWN', 0, 'NONE', ksusmcrncc),  ksusmcrncr, ksusmcropt,  ksusmcrsto FROM
```

x$ksusm WHERE (bitand(ksspaflg, 1) != 0)

VIEW_NAME

VIEW_DEFINITION

GV$STANDBY_APPLY_SNAPSHOT
select INST_ID, SNAPSHOT_TIME, SESSION_ID, THREAD#, RESET_TIMESTAMP, SEQUENCE#,
BLOCK#, APPLIED_SCN, APPLIED_TIME, NEWEST_RESET_TIMESTAMP, NEWEST_ARCHIVED_SEQ#,
NEWEST_TIME, NEWEST_USED, NEWEST_SRL_SEQ#, BLOCKSIZE, APPLY_RATE FROM
x$kcrrptdgstats

GV$STANDBY_LOG
select inst_id, slnum, decode(slpdb,0,'UNASSIGNED',to_number(slpdb)), slthr,
slseq, slsiz*slbsz, decode(slnab, 0, 0, (slnab-1)*slbsz),

VIEW_NAME

VIEW_DEFINITION
---
decode(bitand(slflg,1),0,'NO','YES'),
decode(sign(slseq),0,'UNASSIGNED','ACTIVE'), to_number(sllos),
to_date(sllot,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'), to_number(slnxs),
to_date(slnxt,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian') from x$kccsl

GV$STATISTICS_LEVEL
select inst_id, name, description,   decode(session_status, 0,
'DISABLED', 1, 'ENABLED', 'UNKNOWN'),   decode(system_status, 0,
'DISABLED', 1, 'ENABLED', 'UNKNOWN'),   decode(activation_level, 0,

VIEW_NAME

VIEW_DEFINITION

'BASIC', 1, 'TYPICAL', 'ALL'),   view_name,
decode(session_changeable, 0, 'NO', 'YES')   from x$prmsltyx

GV$STATNAME
select inst_id,indx,ksusdnam,ksusdcls,ksusdhsh from x$ksusd

GV$STREAMS_APPLY_COORDINATOR
select inst_id,sid_knst,serial_knst, applynum_knstacr,
applyname_knstacr,decode(state_knstacr,0,'INITIALIZING',

VIEW_NAME

VIEW_DEFINITION

1,'APPLYING',2,'SHUTTING DOWN
CLEANLY',3,'ABORTING',4,'IDLE'),total_applied_knstacr,
total_waitdeps_knstacr,total_waitcommits_knstacr,total_admin_knstacr,total_assig

```
ned_knstacr,total_received_knstacr,
total_ignored_knstacr,total_rollbacks_knstacr,
total_errors_knstacr,lwm_time_knstacr, lwm_msg_num_knstacr,
lwm_msg_time_knstacr,hwm_time_knstacr, hwm_msg_num_knstacr,
hwm_msg_time_knstacr,startup_time_knstacr, elapsed_schedule_time_knstacr,
elapsed_idle_time_knstacr  from x$knstacr x where type_knst=1 and exists (select

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
1 from v$session s where s.sid=x.sid_knst and s.serial#=x.serial_knst)

GV$STREAMS_APPLY_READER
select inst_id,sid_knst,serial_knst,applynum_knstasl,
applyname_knstasl,decode(state_knstasl,0,'IDLE',8,'DEQUEUE
MESSAGES',10,'SCHEDULE
MESSAGES',15,'INITIALIZING',16,'SPILLING',17,'PAUSED'),total_msg_knstasl,
total_spill_msg_knstasl, last_rcv_time_knstasl,last_rcv_msg_num_knstasl,
last_rcv_msg_time_knstasl, sga_used_knstasl, elapsed_dequeue_time_knstasl,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
elapsed_schedule_time_knstasl,elapsed_spill_time_knstasl,last_browse_num_knstasl
, oldest_scn_num_knstasl,
last_browse_seq_knstasl,last_deq_seq_knstasl,oldest_xid_usn_knstasl,oldest_xid_s
lt_knstasl,oldest_xid_sqn_knstasl,spill_lwm_scn_knstasl,proxy_sid_knstasl,
proxy_serial_knstasl, proxy_spid_knstasl,(SELECT sesstat.ksusestv        FROM
x$ksusd stat, x$ksusesta sesstat, x$ksuse sess WHERE
x.proxy_sid_knstasl=sesstat.ksusenum    AND  x.proxy_sid_knstasl=sess.indx
AND  x.proxy_serial_knstasl=sess.ksuseser    AND  bitand(sess.ksspaflg,1)!=0
AND  bitand(sess.ksuseflg,1)!=0    AND  sesstat.ksusestn = stat.indx    AND

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
bitand(sesstat.ksspaflg,1)!=0    AND  bitand(sesstat.ksuseflg,1)!=0    AND
sesstat.ksusestn<(select ksusgstl from x$ksusgif)   AND  stat.ksusdnam = 'bytes
received via SQL*Net from client')from x$knstasl x where type_knst=7   and
exists (select 1 from v$session s  where s.sid=x.sid_knst and
s.serial#=x.serial_knst)

GV$STREAMS_APPLY_SERVER
select inst_id,sid_knst,serial_knst,applynum_knstasl,
applyname_knstasl,slavid_knstasl,decode(state_knstasl,0,'IDLE',1,'POLL

VIEW_NAME
------------------------------
VIEW_DEFINITION
```

```
--------------------------------------------------------------------------------
SHUTDOWN',2,'RECORD LOW-WATERMARK',3,'ADD PARTITION',4,'DROP
PARTITION',5,'EXECUTE TRANSACTION',6,'WAIT COMMIT',7,'WAIT DEPENDENCY',8,'GET
TRANSACTIONS',9,'WAIT FOR NEXT CHUNK',12,'ROLLBACK TRANSACTION',13,'TRANSACTION
CLEANUP',14,'REQUEST UA SESSION',15,'INITIALIZING'),
xid_usn_knstasl,xid_slt_knstasl,xid_sqn_knstasl,cscn_knstasl,depxid_usn_knstasl,
depxid_slt_knstasl,depxid_sqn_knstasl,depcscn_knstasl,msg_num_knstasl,total_assi
gned_knstasl,total_admin_knstasl,total_rollbacks_knstasl,total_msg_knstasl,
last_apply_time_knstasl,
last_apply_msg_num_knstasl,last_apply_msg_time_knstasl,elapsed_dequeue_time_knst

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
asl, elapsed_apply_time_knstasl from x$knstasl x where type_knst=2 and exists
(select 1 from v$session s where s.sid=x.sid_knst and s.serial#=x.serial_knst)

GV$STREAMS_CAPTURE
SELECT x.inst_id, x.sid_knst, x.serial_knst, x.capnum_knstcap,
x.capname_knstcap, x.logminer_id_knstcap, x.startup_time_knstcap,       case
when (x.state_knstcap = 10 and                 d.loaded = 'ACTIVE')
then d.current_state || ' ' || d.progress            else
DECODE(x.state_knstcap,                 0,  'INITIALIZING',    1,'CAPTURING

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
CHANGES',                2,  'EVALUATING RULE', 3,'ENQUEUING MESSAGE',
4,  'SHUTTING DOWN',   5,'ABORTING',                6,  'CREATING LCR',
7,  DECODE(x.missing_logfile_info_knstcap,                          NULL,
'WAITING FOR DICTIONARY REDO',                          'WAITING FOR
DICTIONARY REDO: ' ||
x.missing_logfile_info_knstcap),                8,
DECODE(x.missing_logfile_info_knstcap,                          NULL,
'WAITING FOR REDO',                          'WAITING FOR REDO: ' ||
x.missing_logfile_info_knstcap),                9,'PAUSED FOR FLOW CONTROL',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
10, 'DICTIONARY INITIALIZATION',                11, 'WAITING FOR APPLY TO
START',                12, 'CONNECTING TO APPLY DATABASE',
13, 'WAITING FOR PROPAGATION TO START')       end,       m.msgs_filtered,
m.msgs_kept, m.msgs_total, x.total_captured_knstcap,
x.recent_time_knstcap, x.recent_msg_num_knstcap,
x.recent_msg_time_knstcap, x.total_messages_created_knstcap,
x.total_full_evaluations_knstcap, x.total_msg_enq_knstcap,
x.enqueue_time_knstcap, x.enqueue_msg_num_knstcap,
x.enqueue_msg_time_knstcap,       DECODE(bitand(x.flags_knstcap, 1), 0,
```

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
upstream.next_scn,            1, downstream.next_scn),
DECODE(bitand(x.flags_knstcap, 1), 0, upstream.next_time,            1,
downstream.next_time),       x.elapsed_capture_time_knstcap,
x.elapsed_rule_time_knstcap,       x.elapsed_enqueue_time_knstcap,
x.elapsed_lcr_time_knstcap, x.elapsed_wait_time_knstcap,
x.elapsed_pause_time_knstcap,        x.state_changed_time_knstcap,
x.appname_knstcap,        x.apply_dblink_knstcap,
x.apply_messages_sent_knstcap,        (SELECT sesstat.ksusestv        FROM
x$ksusd stat, x$ksusesta sesstat, x$ksuse sess        WHERE

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
x.sid_knst=sesstat.ksusenum          AND x.sid_knst=sess.indx          AND
x.serial_knst=sess.ksuseser          AND bitand(sess.ksspaflg,1)!=0
AND bitand(sess.ksuseflg,1)!=0          AND sesstat.ksusestn = stat.indx
AND bitand(sesstat.ksspaflg,1)!=0          AND bitand(sesstat.ksuseflg,1)!=0
AND sesstat.ksusestn<(select ksusgstl from x$ksusgif)          AND
stat.ksusdnam = 'bytes sent via SQL*Net to dblink') FROM   x$knstcap x,
(SELECT session_id,             sum(skipped_filter_calls) msgs_filtered,
sum(kept_filter_calls) msgs_kept,             sum(total_filter_calls)
msgs_total        FROM x$logmnr_process        GROUP BY session_id) m,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
(SELECT last_write_scn next_scn,             last_write_scn_time next_time
FROM x$kcrfws) upstream,       (SELECT session#, max(next_change#) next_scn,
max(next_time) next_time        FROM system.logmnr_log$        GROUP BY
session#) downstream,       x$logmnr_dictionary_load d WHERE  type_knst=8 AND
x.logminer_id_knstcap = m.session_id(+) AND x.logminer_id_knstcap =
d.session_id(+) AND x.logminer_id_knstcap = downstream.session#(+) AND EXISTS
(SELECT 1 FROM v$session s          WHERE s.sid=x.sid_knst AND
s.serial#=x.serial_knst)


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$STREAMS_MESSAGE_TRACKING
SELECT inst_id, tracking_label_knstmt, tag_knstmt, component_name_knstmt,
component_type_knstmt, action_knstmt, action_details_knstmt,
timestamp_knstmt, message_create_time_knstmt, message_number_knstmt,
tracking_id_knstmt, source_database_name_knstmt, object_owner_knstmt,
```

object_name_knstmt, xid_knstmt, command_type_knstmt FROM   x$knstmt

GV$STREAMS_POOL_ADVICE
select inst_id, size_knlarow, round(size_knlarow / basesize_knlarow, 4),

VIEW_NAME

VIEW_DEFINITION
---
spillcnt_knlarow, spilltime_knlarow, unspillcnt_knlarow, unspilltime_knlarow
from x$knlarow

GV$STREAMS_TRANSACTION
SELECT inst_id, strmname_knsttxn, type_knsttxn,   xidusn_knsttxn,
xidslt_knsttxn, xidsqn_knsttxn,   msg_count_knsttxn,
actual_msg_count_knsttxn,   first_msg_time_knsttxn, first_msg_num_knsttxn,
last_msg_time_knsttxn, last_msg_num_knsttxn FROM   x$knsttxn

VIEW_NAME

VIEW_DEFINITION

GV$SUBCACHE
select inst_id,kglnaown, kglnaobj, kglobtyp, kqlfshpn, kqlfscid, kqlfsscc,
kqlfsesp, kqlfsasp, kqlfsusp from x$kqlset

GV$SUBSCR_REGISTRATION_STATS
select inst_id, reg_id, num_ntfns, num_grouping_ntfns, last_ntfn_start_time,
last_ntfn_sent_time, total_emon_latency, emon_server_id,
utl_raw.cast_from_number(all_emon_servers), total_payload_bytes_sent,
num_retries, total_plsql_exec_time/1000000, last_err, last_err_time,

VIEW_NAME

VIEW_DEFINITION
---
last_update_time from x$kkcnrstat

GV$SYSAUX_OCCUPANTS
SELECT inst_id, occ_name_kewxocf, occ_desc_kewxocf,
sch_name_kewxocf, move_proc_kewxocf,   move_desc_kewxocf,
space_usage_kewxocf   FROM   x$kewxocf

GV$SYSMETRIC
SELECT inst_id, begtime, endtime, intsize_csec,   groupid, metricid,

VIEW_NAME

VIEW_DEFINITION
---
name, value, unit   FROM   x$kewmdrmv   WHERE flag1 = 1 AND

groupid in (2,3)

GV$SYSMETRIC_HISTORY
SELECT inst_id, begtime, endtime, intsize_csec, groupid, metricid,
name, value, unit FROM x$kewmdrmv WHERE groupid in (2,3)

GV$SYSMETRIC_SUMMARY
SELECT inst_id, begtime, endtime, intsize_csec, groupid, metricid,

VIEW_NAME

VIEW_DEFINITION
---
name, numintv, max, min, avg, std, unit FROM x$kewmsmdv
WHERE groupid = 2

GV$SYSSTAT
select inst_id,indx,ksusdnam,ksusdcls,ksusgstv,ksusdhsh from x$ksusgsta

GV$SYSTEM_CURSOR_CACHE
select inst_id,kgicsopn,kgicshit,decode(kgicsopn,0,1,kgicshit/kgicsopn) from
x$kgics

VIEW_NAME

VIEW_DEFINITION

GV$SYSTEM_EVENT
select d.inst_id, d.ksllednam, (s.ksleswts_un + s.ksleswts_fg + s.ksleswts_bg),
(s.kslestmo_un + s.kslestmo_fg + s.kslestmo_bg), round((s.kslestim_un +
s.kslestim_fg + s.kslestim_bg)/10000), round((s.kslestim_un + s.kslestim_fg +
s.kslestim_bg)/ (10000 * (s.ksleswts_un + s.ksleswts_fg + s.ksleswts_bg)), 2),
(s.kslestim_un + s.kslestim_fg + s.kslestim_bg), s.ksleswts_fg, s.kslestmo_fg,
round(s.kslestim_fg/10000), round(s.kslestim_fg/decode(s.ksleswts_fg, 0, 1,
10000 * s.ksleswts_fg), 2), s.kslestim_fg, d.ksledhash, d.ksledclassid,

VIEW_NAME

VIEW_DEFINITION
---
d.ksledclass#, d.ksledclass from x$kslei s, x$ksled d where (s.ksleswts_un > 0
or s.ksleswts_fg > 0 or s.ksleswts_bg > 0) and s.indx = d.indx

GV$SYSTEM_FIX_CONTROL
select INST_ID, BUGNO_QKSBGSYROW, VALUE_QKSBGSYROW,
FID_QKSBGSYROW, DESC_QKSBGSYROW, OFE_QKSBGSYROW,
EVENT_QKSBGSYROW, ISDEFAULT_QKSBGSYROW from x$qksbgsys

GV$SYSTEM_PARAMETER

VIEW_NAME

```
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select x.inst_id,x.indx+1,ksppinm,ksppity,ksppstvl, ksppstdvl, ksppstdf,
decode(bitand(ksppiflg/256,1),1,'TRUE','FALSE'),
decode(bitand(ksppiflg/65536,3),1,'IMMEDIATE',2,'DEFERRED',
3,'IMMEDIATE','FALSE'),  decode(bitand(ksppiflg,4),4,'FALSE',
decode(bitand(ksppiflg/65536,3), 0, 'FALSE', 'TRUE')),
decode(bitand(ksppstvf,7),1,'MODIFIED','FALSE'),
decode(bitand(ksppstvf,2),2,'TRUE','FALSE'),  decode(bitand(ksppilrmflg/64, 1),
1, 'TRUE', 'FALSE'),  decode(bitand(ksppilrmflg/268435456, 1), 1, 'TRUE',
'FALSE'),  ksppdesc, ksppstcmnt, ksppihash from x$ksppi x, x$ksppsv y  where

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
(x.indx = y.indx) and  ((translate(ksppinm,'_','#') not like '##%') and
((translate(ksppinm,'_','#') not like '#%')      or (ksppstdf = 'FALSE') or
(bitand(ksppstvf,5) > 0)))

GV$SYSTEM_PARAMETER2
select x.inst_id,kspftctxpn,ksppinm,ksppity,kspftctxvl,  kspftctxdvl,
kspftctxdf,  decode(bitand(ksppiflg/256,1),1,'TRUE','FALSE'),
decode(bitand(ksppiflg/65536,3),1,'IMMEDIATE',2,'DEFERRED',
3,'IMMEDIATE','FALSE'),  decode(bitand(ksppiflg,4),4,'FALSE',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(bitand(ksppiflg/65536,3), 0, 'FALSE', 'TRUE')),
decode(bitand(kspftctxvf,7),1,'MODIFIED','FALSE'),
decode(bitand(kspftctxvf,2),2,'TRUE','FALSE'),  decode(bitand(ksppilrmflg/64,
1), 1, 'TRUE', 'FALSE'),  decode(bitand(ksppilrmflg/268435456, 1), 1, 'TRUE',
'FALSE'),  ksppdesc, kspftctxvn,  kspftctxct from x$ksppi x, x$ksppsv2 y  where
((x.indx+1) = kspftctxpn) and  ((translate(ksppinm,'_','#') not like '##%') and
(translate(ksppinm,'_','#') not like '#%'     or (kspftctxdf = 'FALSE') or
(bitand(kspftctxvf,5) > 0)))


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$SYSTEM_PARAMETER3
select x.inst_id,x.indx+1,ksppinm,ksppstdvl from x$ksppi x, x$ksppsv y  where
(x.indx = y.indx)

GV$SYSTEM_PARAMETER4
select x.inst_id,kspftctxsid,kspftctxpn,ksppinm,ksppity,kspftctxdvl,
kspftctxvn,kspftctxct  from x$ksppi x, x$ksppsv2 y  where ((x.indx+1) =
```

kspftctxpn) and (bitand(ksppilrmflg,64)!=64) and ((kspftctxdf = 'FALSE') or (bitand(kspftctxvf,8) = 8))

VIEW_NAME

VIEW_DEFINITION

GV$SYSTEM_WAIT_CLASS
select s.inst_id, s.kslscsclsid, s.kslscscls, s.kslscsclsname, (s.kslscswts_un + s.kslscswts_fg + s.kslscswts_bg), round((s.kslscstim_un + s.kslscstim_fg + s.kslscstim_bg)/10000), s.kslscswts_fg, round(s.kslscstim_fg/10000) from x$kslscs s where s.kslscswts_un > 0 or s.kslscswts_fg > 0 or s.kslscswts_un > 0

GV$SYS_OPTIMIZER_ENV
select INST_ID, PNUM_QKSCESYROW,

VIEW_NAME

VIEW_DEFINITION

PNAME_QKSCESYROW, FID_QKSCESYROW,
decode(bitand(FLAGS_QKSCESYROW, 2), 0, 'NO', 'YES'),
PVALUE_QKSCESYROW,
DEFPVALUE_QKSCESYROW from X$QKSCESYS
where bitand(FLAGS_QKSCESYROW, 8) = 0 and
(bitand(FLAGS_QKSCESYROW, 4) = 0 or
bitand(FLAGS_QKSCESYROW, 2) = 0)

GV$SYS_TIME_MODEL

VIEW_NAME

VIEW_DEFINITION
---
select map.inst_id, map.extid, map.sname, sysv.kewsval from x$kewssmap map, x$kewssysv sysv where map.offst = sysv.indx and map.aggid = 1 and (map.stype = 2 or map.stype = 3)

GV$TABLESPACE
select inst_id,tstsn,tsnam, decode(bitand(tsflg, 1+2), 1, 'NO', 2,'NO','YES'), decode(bitand(tsflg, 4), 4,'YES','NO'), decode(bitand(tsflg, 8), 8,'NO','YES'), decode(bitand(tsflg, 16+32), 16, 'ON', 32, 'OFF', to_char(null)) from x$kccts where tstsn != -1

VIEW_NAME

VIEW_DEFINITION

GV$TEMPFILE
select tf.inst_id, tf.tfnum, to_number(tf.tfcrc_scn),

```
to_date(tf.tfcrc_tim,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'), tf.tftsn,
tf.tfrfn, decode(bitand(tf.tfsta, 2),0,'OFFLINE',2,'ONLINE','UNKNOWN'),
decode(bitand(tf.tfsta, 12), 0,'DISABLED',4, 'READ ONLY', 12, 'READ WRITE',
'UNKNOWN'), fh.fhtmpfsz*tf.tfbsz, fh.fhtmpfsz,  tf.tfcsz*tf.tfbsz,tf.tfbsz,
fn.fnnam  from x$kcctf tf, x$kccfn fn, x$kcvfhtmp fh  where fn.fnfno=tf.tfnum
and fn.fnfno=fh.htmpxfil and tf.tffnh=fn.fnnum  and tf.tfdup!=0 and

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
bitand(tf.tfsta, 32) <> 32  and fn.fntyp=7 and fn.fnnam is not null

GV$TEMPORARY_LOBS
select kdlt.inst_id, kdlt.kdltsno, sum(kdlt.kdltctmp),
sum(kdlt.kdltnctmp), abs.count                          from X$KDLT
kdlt, X$ABSTRACT_LOB abs                      group by kdlt.inst_id,
kdlt.kdltsno, abs.count              order by kdltsno

GV$TEMPSTAT

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select k.inst_id,k.kcftiofno,k.kcftiopyr,k.kcftiopyw,k.kcftiopbr,k.kcftiopbw,
k.kcftiosbr,k.kcftioprt,k.kcftiopwt,k.kcftiosbt,k.kcftioavg,k.kcftiolst,
k.kcftiomin,k.kcftiormx,k.kcftiowmx from x$kcftio k,x$kcctf f where f.tfdup <> 0
and f.tfnum=k.kcftiofno

GV$TEMP_CACHE_TRANSFER
select x.inst_id, kcftiofno,       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  from
x$kcftio x, x$kcctf tf        where x.kcftiofno = tf.tfnum


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$TEMP_EXTENT_MAP
select /*+ ordered use_nl(me) */ me.inst_id, ts.name, me.ktftmetfno,
me.ktftmebno, me.ktftmeblks*ts.blocksize, me.ktftmeblks, me.ktftmeinst,
me.ktftmefno from ts$ ts, x$ktftme me where ts.contents$ = 1 and ts.bitmapped <>
0 and ts.online$ = 1 and ts.ts# = me.ktftmetsn

GV$TEMP_EXTENT_POOL
select /*+ ordered use_nl(fc) */ fc.inst_id, ts.name, fc.ktstfctfno,
fc.ktstfcec, fc.ktstfceu, fc.ktstfcbc, fc.ktstfcbu, fc.ktstfcbc*ts.blocksize,

VIEW_NAME
------------------------------
VIEW_DEFINITION
```

```
--------------------------------------------------------------------------------
fc.ktstfcbu*ts.blocksize, fc.ktstfcfno from ts$ ts, x$ktstfc fc where
ts.contents$ = 1 and ts.bitmapped <> 0 and ts.online$ = 1 and ts.ts# =
fc.ktstfctsn

GV$TEMP_PING
select x.inst_id, kcftiofno,  0, 0, 0, 0, 0, 0, 0, 0,      0, 0, 0, 0, 0, 0, 0,
0,      0, 0, 0, 0, 0, 0, 0, 0     from x$kcftio x, x$kcctf tf   where
x.kcftiofno = tf.tfnum


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$TEMP_SPACE_HEADER
select /*+ ordered use_nl(hc) */ hc.inst_id, ts.name, hc.ktfthctfno,
(hc.ktfthcsz - hc.ktfthcfree)*ts.blocksize, (hc.ktfthcsz - hc.ktfthcfree),
hc.ktfthcfree*ts.blocksize, hc.ktfthcfree, hc.ktfthcfno from ts$ ts, x$ktfthc hc
where ts.contents$ = 1 and ts.bitmapped <> 0 and ts.online$ = 1 and ts.ts# =
hc.ktfthctsn and hc.ktfthccval = 0

GV$THREAD
select rt.inst_id,rtnum,decode(bitand(rtsta,1),1,'OPEN','CLOSED'),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(bitand(rtsta,6),0,'DISABLED',2,'PRIVATE',6,'PUBLIC','UNKNOWN'),
rtnlf,tirsid,to_date(rtots,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),
rtcln,rtseq, to_number(rtckp_scn), to_date(rtckp_tim,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'), to_number(rtenb), to_date(rtets,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'), to_number(rtdis), to_date(rtdit,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'), cpodr_seq, cpodr_bno, to_number(cpods),
to_date(cpodt,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian') from x$kccrt rt,
x$kcctir tr, x$kcccp  where rtnlf != 0 and tr.inst_id = rt.inst_id  and tirnum =
rtnum and cptno = rtnum

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

GV$THRESHOLD_TYPES
SELECT t.inst_id, mid_kelrtd, gid_kelrtd, opmask_kelrtd,
typnam_keltosd, alrtid_kelrtd, valtype_kelrtd         FROM x$kelrtd t,
x$keltosd o         WHERE typid_keltosd = objtype_kelrtd

GV$TIMER
select inst_id,ksutmtim from x$ksutm
```

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$TIMEZONE_FILE
select FILENAME, VERSION from X$TIMEZONE_FILE

GV$TIMEZONE_NAMES
select TZNAME, TZABBREV from X$TIMEZONE_NAMES

GV$TRANSACTION
select inst_id,ktcxbxba,kxidusn,kxidslt,kxidsqn,ktcxbkfn,kubablk,
kubaseq,kubarec,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(ktcxbsta,0,'IDLE',1,'COLLECTING',2,'PREPARED',3,'COMMITTED',
4,'HEURISTIC ABORT',5,'HEURISTIC COMMIT',              6,'HEURISTIC
DAMAGE',7,'TIMEOUT',9,'INACTIVE',              10,'ACTIVE',11,'PTX
PREPARED',12,'PTX COMMITTED',              'UNKNOWN'),
ktcxbstm,ktcxbssb,ktcxbssw, ktcxbsen,ktcxbsfl,ktcxbsbk,ktcxbssq,ktcxbsrc,
ktcxbses,ktcxbflg, decode(bitand(ktcxbflg,16),0,'NO','YES'),
decode(bitand(ktcxbflg,32),0,'NO','YES'),
decode(bitand(ktcxbflg,64),0,'NO','YES'),
decode(bitand(ktcxbflg,8388608),0,'NO','YES'), ktcxbnam,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
ktcxbpus,ktcxbpsl,ktcxbpsq, ktcxbpxu,ktcxbpxs,ktcxbpxq, ktcxbdsb, ktcxbdsw,
ktcxbubk,ktcxburc,ktcxblio,ktcxbpio,ktcxbcrg,ktcxbcrc,
to_date(ktcxbstm,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'), ktcxbdsb,
ktcxbdsw,  ktcxbssc, ktcxbdsc, ktcxbxid, ktcxbpid, ktcxbpxi  from x$ktcxb where
bitand(ksspaflg,1)!=0 and bitand(ktcxbflg,2)!=0

GV$TRANSACTION_ENQUEUE
select  s.inst_id,l.ktcxbxba,l.ktcxblkp,s.ksusenum,r.ksqrsidt,r.ksqrsid1,
r.ksqrsid2, l.ksqlkmod, l.ksqlkreq,l.ksqlkctim,l.ksqlklblk from x$ktcxb

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
l,x$ksuse s,x$ksqrs r where l.ksqlkses=s.addr and  bitand(l.ksspaflg,1)!=0 and
(l.ksqlkmod!=0 or l.ksqlkreq!=0) and  l.ksqlkres=r.addr

GV$TRANSPORTABLE_PLATFORM
SELECT INST_ID, PLATFORM_ID, PLATFORM_NAME,         decode(endian_format,
```

```
1,'Big' ,0,'Little','UNKNOWN FORMAT')          FROM  x$kcpxpl

GV$TSM_SESSIONS
select  inst_id,  indx,  ksusmser,  decode(ksusmsta,        0,  'NONE',

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
1,  'SELECTED',        2,  'COMMITED SELECT',        3,  'READY FOR PREPARE',
4,  'PREPARED',        5,  'READY FOR SWITCH',        6,  'SWITCHED',
7,  'FAILED',        8,  'READY FOR STATE TRANSFER',        9,  'IN STATE
TRANSFER',        10, 'END OF STATE TRANSFER',        'UNKNOWN'),
decode(bitand(ksusmflg, 1),        1, 'NO',        decode(ksusmbnd,
0, decode(bitand(ksusmflg, 8),                    8, 'NO',
'YES'),              'YES')),  decode(bitand(ksusmflg, 1), 1, 'NO', 'YES'),
decode(bitand(ksusmflg, 8), 8, 'NO', 'YES'),  decode(bitand(ksusmflg, 16), 16,
'YES', 'NO'),  ksusmnmr,  ksusmnmi,  ksusmntr,  ksusmnti,  decode(ksusmbnd, 0,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
'NEVER', 1, 'CALL', 2, 'TRANSACTION', 3, 'APPLICATION',        'UNKNOWN'),
decode(bitand(ksusmflg, 4), 4, 'YES', 'NO'),  ksusmcst,  ksusmdst,  ksusmnrt,
ksusmbcm,  ksusmstm,  ksusmseq   from x$ksusm  where (bitand(ksspaflg, 1) != 0)
and   (bitand(ksusmsfl, 1) != 0)

GV$TYPE_SIZE
select inst_id,kqfszcom,kqfsztyp,kqfszdsc,kqfszsiz from x$kqfsz

GV$UNDOSTAT

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select inst_id,      to_date(KTUSMSTRBEGTIME,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),      to_date(KTUSMSTRENDTIME,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),      KTUSMSTTSN, KTUSMSTUSU, KTUSMSTTCT,
KTUSMSTMQL, KTUSMSTRMQI, KTUSMSTMTC,      KTUSMSTUAC, KTUSMSTUBS, KTUSMSTUBR,
KTUSMSTXAC, KTUSMSTXBS,  KTUSMSTXBR,      KTUSMSTSOC, KTUSMSTOOS, KTUSMSTABK,
KTUSMSTUBK, KTUSMSTEBK,  KTUSMSTTUR      from X$KTUSMST

GV$VERSION
select inst_id, banner from x$version

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
```

GV$VPD_POLICY
select c.inst_id,c.kglhdadr,c.kglhdpar,c.kglnahsh, c.kglobt03,
c.kglobt09, p.kzrtpdow,p.kzrtpdon,p.kzrtpdgp,p.kzrtpdpy,p.kzrtpdpo,
p.kzrtpdtx from x$kglcursor_child c, x$kzrtpd p where c.kglhdpar = p.kzrtpdpa
and c.kglhdadr = p.kzrtpdad

GV$WAITCLASSMETRIC
SELECT inst_id, begtime, endtime, intsize_csec, wait#, wait_id,

VIEW_NAME

VIEW_DEFINITION
---
average_waiter_count, dbtime_in_wait, time_waited, wait_count
FROM x$kewmevmv WHERE flag1 = 1 AND GROUPID = 1

GV$WAITCLASSMETRIC_HISTORY
SELECT inst_id, begtime, endtime, intsize_csec, wait#, wait_id,
average_waiter_count, dbtime_in_wait, time_waited, wait_count
FROM x$kewmevmv WHERE GROUPID = 1

GV$WAITSTAT

VIEW_NAME

VIEW_DEFINITION
---
select inst_id,decode(indx,1,'data block',2,'sort block',3,'save undo block',
4,'segment header',5,'save undo header',6,'free list',7,'extent map', 8,'1st
level bmb',9,'2nd level bmb',10,'3rd level bmb', 11,'bitmap block',12,'bitmap
index block',13,'file header block',14,'unused', 15,'system undo
header',16,'system undo block', 17,'undo header',18,'undo block'), count,time
from x$kcbwait where indx!=0

GV$WALLET
SELECT INST_ID, CERTID, CERTDN, CERTSERIAL, CERTISSUER, KEYSIZE,

VIEW_NAME

VIEW_DEFINITION

CERTSTATUS FROM X$KZEKMFVW

GV$WORKLOAD_REPLAY_THREAD
SELECT inst_id, clock, next_ticker, sid, serial#, spid, logon_user,
logon_time, event, event_id, event#, p1text, p1, p2text, p2,
p3text, p3, wait_for_scn, file_id, call_counter, dependent_scn,
statement_scn, commit_wait_scn, post_commit_scn, action_type,
session_type, wrc_id, file_name, skip_it, dirty_buffers, dbtime,
network_time, think_time, time_gain, time_loss, user_calls,

VIEW_NAME

VIEW_DEFINITION
---
client_os_user, client_host, client_pid, program FROM X$KECPRT

GV$XML_AUDIT_TRAIL
select INST_ID, AUDIT_TYPE, SESSION_ID, PROXY_SESSIONID, STATEMENTID, ENTRYID,
EXTENDED_TIMESTAMP, GLOBAL_UID, DB_USER, CLIENTIDENTIFIER, EXT_NAME,
OS_USER, OS_HOST, OS_PROCESS, TERMINAL, INSTANCE_NUMBER, OBJECT_SCHEMA,
OBJECT_NAME, POLICY_NAME, NEW_OWNER, NEW_NAME, ACTION, STATEMENT_TYPE,
TRANSACTIONID, RETURNCODE, SCN, COMMENT_TEXT, AUTH_PRIVILEGES, GRANTEE,
PRIV_USED, SES_ACTIONS, OS_PRIVILEGE, ECONTEXT_ID, SQL_BIND, SQL_TEXT,

VIEW_NAME

VIEW_DEFINITION

OBJ_EDITION_NAME from X$XML_AUDIT_TRAIL

GV$_LOCK
select USERENV('Instance'),laddr,kaddr,saddr,raddr,lmode,request,ctime, block
from v$_lock1 union all select
inst_id,addr,ksqlkadr,ksqlkses,ksqlkres,ksqlkmod,ksqlkreq, ksqlkctim,ksqlklblk
from x$ktadm where bitand(kssobflg,1)!=0 and (ksqlkmod!=0 or ksqlkreq!=0) union
all select inst_id,addr,ksqlkadr,ksqlkses,ksqlkres,ksqlkmod,ksqlkreq,
ksqlkctim,ksqlklblk from x$ktatrfil where bitand(kssobflg,1)!=0 and (ksqlkmod!=0

VIEW_NAME

VIEW_DEFINITION
---
or ksqlkreq!=0) union all select
inst_id,addr,ksqlkadr,ksqlkses,ksqlkres,ksqlkmod,ksqlkreq, ksqlkctim,ksqlklblk
from x$ktatrfsl where bitand(kssobflg,1)!=0 and (ksqlkmod!=0 or ksqlkreq!=0)
union all select inst_id,addr,ksqlkadr,ksqlkses,ksqlkres,ksqlkmod,ksqlkreq,
ksqlkctim,ksqlklblk from x$ktatl where bitand(kssobflg,1)!=0 and (ksqlkmod!=0 or
ksqlkreq!=0) union all select
inst_id,addr,ksqlkadr,ksqlkses,ksqlkres,ksqlkmod,ksqlkreq, ksqlkctim,ksqlklblk
from x$ktstusc where bitand(kssobflg,1)!=0 and (ksqlkmod!=0 or ksqlkreq!=0)
union all select inst_id,addr,ksqlkadr,ksqlkses,ksqlkres,ksqlkmod,ksqlkreq,

VIEW_NAME

VIEW_DEFINITION
---
ksqlkctim,ksqlklblk from x$ktstuss where bitand(kssobflg,1)!=0 and (ksqlkmod!=0
or ksqlkreq!=0) union all select
inst_id,addr,ksqlkadr,ksqlkses,ksqlkres,ksqlkmod,ksqlkreq, ksqlkctim,ksqlklblk
from x$ktstusg where bitand(kssobflg,1)!=0 and (ksqlkmod!=0 or ksqlkreq!=0)
union all select inst_id,ktcxbxba,ktcxblkp,ksqlkses,ksqlkres,ksqlkmod,ksqlkreq,
ksqlkctim,ksqlklblk from x$ktcxb where bitand(ksspaflg,1)!=0 and (ksqlkmod!=0 or
ksqlkreq!=0)

GV$_LOCK1

VIEW_NAME

VIEW_DEFINITION
---
select inst_id,addr,ksqlkadr,ksqlkses,ksqlkres,ksqlkmod,ksqlkreq, ksqlkctim,
ksqlklblk from x$kdnssf where bitand(kssobflg,1)!=0 and (ksqlkmod!=0 or
ksqlkreq!=0) union all select
inst_id,addr,ksqlkadr,ksqlkses,ksqlkres,ksqlkmod,ksqlkreq, ksqlkctim, ksqlklblk
from x$ksqeq where bitand(kssobflg,1)!=0 and (ksqlkmod!=0 or ksqlkreq!=0)

GV$_RESUMABLE2
select inst_id, ktrsfaddr, ktrsfsid, decode (bitand(ktrsfflg, 1), 0, 'NO',
'YES'), decode (ktrsfsta, 0, 'NORMAL', 1, 'SUSPENDED', 2, 'TIMEOUT', 3, 'ERROR',

VIEW_NAME

VIEW_DEFINITION

4, 'ABORTED', ''), ktrsftmo, ktrsfspt, ktrsfrst, ktrsfnam, ktrsferr, ktrsfep1,
ktrsfep2, ktrsfep3, ktrsfep4, ktrsfep5, ktrsfems, ktrsfobj, ktrsftyp from
x$ktrso

GV$_SEQUENCES
select inst_id,
KGLNAOWN,KGLNAOBJ,KGLOBT08,decode(bitand(KGLOBT00,1),0,'N','Y'),decode(bitand(KG
LOBT00,2),0,'N','Y'),decode(bitand(KGLOBT00,16),0,'N','Y'),KGLOBTN0,KGLOBTN2,KGL
OBTN3,KGLOBTN1,decode(bitand(KGLOBT09,1),0,'N','Y'),decode(bitand(KGLOBT09,2),0,

VIEW_NAME

VIEW_DEFINITION

'N','Y'),KGLOBTN4,KGLOBTN5,decode(KGLOBT10,1,'Y','N'),decode(KGLOBT10,1,KGLOBT02
,null)from X$KGLOB where KGLOBTYP = 6 and KGLOBT11 = 1

O$SQL_BIND_CAPTURE
select ADDRESS, HASH_VALUE, SQL_ID, CHILD_ADDRESS, CHILD_NUMBER,
NAME, POSITION,
DUP_POSITION, DATATYPE, DATATYPE_STRING, CHARACTER_SID,
PRECISION, SCALE, MAX_LENGTH, WAS_CAPTURED, LAST_CAPTURED,
VALUE_STRING, VALUE_ANYDATA from go$sql_bind_capture

VIEW_NAME

VIEW_DEFINITION
---
where inst_id = USERENV('Instance')

V$ACCESS

select SID , OWNER , OBJECT , TYPE from GV$ACCESS where inst_id =
USERENV('Instance')

V$ACTIVE_INSTANCES
select INST_NUMBER , INST_NAME from GV$ACTIVE_INSTANCES where inst_id =
USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$ACTIVE_SERVICES
select SERVICE_ID, NAME, NAME_HASH, NETWORK_NAME, CREATION_DATE,
CREATION_DATE_HASH, GOAL, DTP, BLOCKED, AQ_HA_NOTIFICATION, CLB_GOAL from
GV$ACTIVE_SERVICES where inst_id = USERENV('Instance')

V$ACTIVE_SESSION_HISTORY
SELECT sample_id, sample_time, session_id, session_serial#, session_type, flags,
user_id, sql_id, sql_child_number, sql_opcode, force_matching_signature,

VIEW_NAME

VIEW_DEFINITION
----
top_level_sql_id, top_level_sql_opcode, sql_plan_hash_value, sql_plan_line_id,
sql_plan_operation, sql_plan_options, sql_exec_id, sql_exec_start,
plsql_entry_object_id, plsql_entry_subprogram_id, plsql_object_id,
plsql_subprogram_id, qc_instance_id, qc_session_id, qc_session_serial#, event,
event_id, event#, seq#, p1text, p1, p2text, p2, p3text, p3, wait_class,
wait_class_id, wait_time, session_state, time_waited, blocking_session_status,
blocking_session, blocking_session_serial#, current_obj#, current_file#,
current_block#, current_row#, consumer_group_id, xid, remote_instance#,
in_connection_mgmt, in_parse, in_hard_parse, in_sql_execution,

VIEW_NAME

VIEW_DEFINITION
----
in_plsql_execution, in_plsql_rpc, in_plsql_compilation, in_java_execution,
in_bind, in_cursor_close, service_hash, program, module, action, client_id FROM
GV$ACTIVE_SESSION_HISTORY WHERE inst_id = USERENV('INSTANCE')

V$ACTIVE_SESS_POOL_MTH
select name from gv$active_sess_pool_mth   where inst_id =
userenv('instance')

V$ADVISOR_PROGRESS

VIEW_NAME

VIEW_DEFINITION

```
--------------------------------------------------------------------------------
select SID, SERIAL#, USERNAME, OPNAME, ADVISOR_NAME, TASK_ID,
TARGET_DESC, SOFAR, TOTALWORK, UNITS,
BENEFIT_SOFAR, BENEFIT_MAX, FINDINGS,  RECOMMENDATIONS,
TIME_REMAINING, START_TIME,
LAST_UPDATE_TIME,
ELAPSED_SECONDS,
ADVISOR_METRIC1, METRIC1_DESC,
EXECUTION_TYPE                                  from
GV$ADVISOR_PROGRESS                                     where

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
inst_id = USERENV('Instance')

V$ALERT_TYPES
SELECT reason_id, object_type, type, group_name, scope,
internal_metric_category, internal_metric_name          FROM gv$alert_types
WHERE inst_id = USERENV('INSTANCE')

V$AQ1
select  QID , WAITING, READY, EXPIRED, TOTAL_CONSUMERS,         TOTAL_WAIT,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
AVERAGE_WAIT         from GV$AQ1

V$ARCHIVE
select  GROUP# , THREAD# , SEQUENCE# , ISCURRENT , "CURRENT" , FIRST_CHANGE#
from GV$ARCHIVE where inst_id = USERENV('Instance')

V$ARCHIVED_LOG
select  RECID , STAMP , NAME , DEST_ID , THREAD# , SEQUENCE# ,
RESETLOGS_CHANGE# , RESETLOGS_TIME , RESETLOGS_ID ,  FIRST_CHANGE# , FIRST_TIME

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
,  NEXT_CHANGE# , NEXT_TIME , BLOCKS , BLOCK_SIZE , CREATOR, REGISTRAR ,
STANDBY_DEST , ARCHIVED , APPLIED , DELETED , STATUS , COMPLETION_TIME ,
DICTIONARY_BEGIN , DICTIONARY_END , END_OF_REDO, BACKUP_COUNT ,
ARCHIVAL_THREAD#, ACTIVATION#, IS_RECOVERY_DEST_FILE, COMPRESSED, FAL,
END_OF_REDO_TYPE, BACKED_BY_VSS from GV$ARCHIVED_LOG where inst_id =
USERENV('Instance')

V$ARCHIVE_DEST
select DEST_ID,DEST_NAME, STATUS, BINDING, NAME_SPACE, TARGET, ARCHIVER,
```

VIEW_NAME

VIEW_DEFINITION

SCHEDULE, DESTINATION, LOG_SEQUENCE, REOPEN_SECS, DELAY_MINS, MAX_CONNECTIONS,
NET_TIMEOUT, PROCESS, REGISTER, FAIL_DATE, FAIL_SEQUENCE, FAIL_BLOCK,
FAILURE_COUNT, MAX_FAILURE, ERROR, ALTERNATE, DEPENDENCY, REMOTE_TEMPLATE,
QUOTA_SIZE, QUOTA_USED, MOUNTID, TRANSMIT_MODE, ASYNC_BLOCKS, AFFIRM, TYPE,
VALID_NOW, VALID_TYPE, VALID_ROLE, DB_UNIQUE_NAME, VERIFY, COMPRESSION from
GV$ARCHIVE_DEST where inst_id = USERENV('Instance')

V$ARCHIVE_DEST_STATUS
select DEST_ID,DEST_NAME, STATUS, TYPE, DATABASE_MODE, RECOVERY_MODE,

VIEW_NAME

VIEW_DEFINITION

PROTECTION_MODE, DESTINATION, STANDBY_LOGFILE_COUNT, STANDBY_LOGFILE_ACTIVE,
ARCHIVED_THREAD#, ARCHIVED_SEQ#, APPLIED_THREAD#, APPLIED_SEQ#, ERROR, SRL,
DB_UNIQUE_NAME, SYNCHRONIZATION_STATUS, SYNCHRONIZED from
GV$ARCHIVE_DEST_STATUS where inst_id = USERENV('Instance')

V$ARCHIVE_GAP
select THREAD# , LOW_SEQUENCE# , HIGH_SEQUENCE# from GV$ARCHIVE_GAP where
inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$ARCHIVE_PROCESSES
select PROCESS, STATUS, LOG_SEQUENCE, STATE from GV$ARCHIVE_PROCESSES where
inst_id = USERENV('Instance')

V$ASM_ALIAS
select name, group_number, file_number, file_incarnation, alias_index,
alias_incarnation, parent_index, reference_index, alias_directory,
system_created from gv$asm_alias where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$ASM_ATTRIBUTE
select name, value, group_number, attribute_index, attribute_incarnation,
read_only, system_created from gv$asm_attribute where inst_id =
USERENV('Instance')

V$ASM_CLIENT
select group_number, instance_name, db_name, status, software_version,
compatible_version from gv$asm_client where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$ASM_DISK
select group_number, disk_number, compound_index, incarnation, mount_status,
header_status, mode_status, state, redundancy, library, os_mb, total_mb,
free_mb, name, failgroup, label, path, udid, product, create_date, mount_date,
repair_timer, reads, writes, read_errs, write_errs, read_time, write_time,
bytes_read, bytes_written, preferred_read from gv$asm_disk where inst_id =
USERENV('Instance')

V$ASM_DISKGROUP

VIEW_NAME

VIEW_DEFINITION
---
select group_number, name, sector_size, block_size, allocation_unit_size,
state, type, total_mb, free_mb, required_mirror_free_mb, usable_file_mb,
offline_disks, compatibility, database_compatibility from gv$asm_diskgroup
where inst_id = USERENV('Instance')

V$ASM_DISKGROUP_STAT
select group_number, name, sector_size, block_size, allocation_unit_size,
state, type, total_mb, free_mb, required_mirror_free_mb, usable_file_mb,
offline_disks, compatibility, database_compatibility from

VIEW_NAME

VIEW_DEFINITION
---
gv$asm_diskgroup_stat where inst_id = USERENV('Instance')

V$ASM_DISK_IOSTAT
select instname, dbname, group_number, disk_number, failgroup, reads,
writes, read_errs, write_errs, read_time, write_time, bytes_read, bytes_written
from gv$asm_disk_iostat where inst_id = USERENV('Instance')

V$ASM_DISK_STAT
select group_number, disk_number, compound_index, incarnation, mount_status,

VIEW_NAME

VIEW_DEFINITION
---
header_status, mode_status, state, redundancy, library, os_mb, total_mb,

free_mb, name, failgroup, label, path, udid, product, create_date, mount_date,
repair_timer, reads, writes, read_errs, write_errs, read_time, write_time,
bytes_read, bytes_written, preferred_read from gv$asm_disk_stat where
inst_id = USERENV('Instance')

V$ASM_FILE
select group_number, file_number, compound_index, incarnation, block_size,
blocks, bytes, space, type, redundancy, striped, creation_date,

VIEW_NAME

VIEW_DEFINITION
----
modification_date, redundancy_lowered from gv$asm_file where inst_id =
USERENV('Instance')

V$ASM_OPERATION
select group_number, operation, state, power, actual, sofar, est_work,
est_rate, est_minutes, error_code from gv$asm_operation where inst_id =
USERENV('Instance')

V$ASM_TEMPLATE

VIEW_NAME

VIEW_DEFINITION
----
select group_number, entry_number, redundancy, stripe, system, name from
gv$asm_template where inst_id = USERENV('Instance')

V$AW_AGGREGATE_OP
select name_xsagopft as name, desc_xsagopft as longname, case when
weight_xsagopft >= 0 then weight_xsagopft else null end as default_weight from
x$xsagop where not bitand(flags_xsagopft, 32) = 0

V$AW_ALLOCATE_OP

VIEW_NAME

VIEW_DEFINITION
----
select name_xsagopft as name, desc_xsagopft as longname from x$xsagop where not
bitand(flags_xsagopft, 64) = 0

V$AW_CALC
select SESSION_ID, AGGREGATE_CACHE_HITS, AGGREGATE_CACHE_MISSES,
SESSION_CACHE_HITS, SESSION_CACHE_MISSES, POOL_HITS, POOL_MISSES,
POOL_NEW_PAGES, POOL_RECLAIMED_PAGES, CACHE_WRITES, POOL_SIZE,
CURR_DML_COMMAND,
PREV_DML_COMMAND, AGGR_FUNC_LOGICAL_NA, AGGR_FUNC_PRECOMPUTE,
AGGR_FUNC_CALCS
from gv$aw_calc where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$AW_LONGOPS
select SESSION_ID, CURSOR_NAME, COMMAND, STATUS, ROWS_PROCESSED, START_TIME from gv$aw_longops where inst_id = USERENV('Instance')

V$AW_OLAP
select session_id, aw_number, attach_mode, generation, temp_space_pages, temp_space_reads, lob_reads, pool_changed_pages, pool_unchanged_pages from gv$aw_olap where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$AW_SESSION_INFO
select SESSION_ID, CLIENT_TYPE, SESSION_STATE, SESSION_HANDLE, USERID, TOTAL_TRANSACTION, TRANSACTION_TIME, TOTAL_TRANSACTION_TIME, AVERAGE_TRANSACTION_TIME, TRANSACTION_CPU_TIME, TOTAL_TRANSACTION_CPU_TIME, AVERAGE_TRANSACTION_CPU_TIME from gv$aw_session_info where inst_id = USERENV('Instance')

V$BACKUP

VIEW_NAME

VIEW_DEFINITION
---
select FILE# , STATUS , CHANGE# , TIME from GV$BACKUP where inst_id = USERENV('Instance')

V$BACKUP_ARCHIVELOG_DETAILS
select a.*, sys.dbms_rcvman.num2displaysize(filesize) filesize_display from (select unique 'BACKUPSET' btype, b.recid btype_key, b.session_recid session_key, b.session_recid, b.session_stamp, a.set_stamp id1, b.set_count id2, thread#, sequence#, resetlogs_change#, resetlogs_time, first_change#, first_time, next_change#,

VIEW_NAME

VIEW_DEFINITION
---
next_time, (blocks+1)*a.block_size filesize, case when b.compression_ratio>1 then b.compression_ratio else 1 end compression_ratio from v$backup_redolog a, v$backup_set_details b where a.set_stamp = b.set_stamp and a.set_count = b.set_count union select unique 'PROXYCOPY', a.recid btype_key, session_recid session_key,

session_recid, session_stamp, a.recid, a.stamp, thread#,
sequence#, resetlogs_change#, resetlogs_time, first_change#, first_time,
next_change#, next_time, (blocks+1)*block_size filesize, 1 from
v$proxy_archivedlog a, v$rman_status b, (select /*+ no_merge */

VIEW_NAME

VIEW_DEFINITION
----
sys.dbms_rcvman.sv_getsessionkey skey from dual)c, (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionfromTimeRange fTime from dual) d, (select
/*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime from
dual) e where a.status = 'A' and a.rman_status_recid = b.recid
(+) and a.rman_status_stamp = b.stamp (+) and (c.skey is
null or c.skey = b.session_recid) and (d.fTime is null or d.fTime <=
b.start_time) and (e.uTime is null or e.uTime >= b.end_time))a

V$BACKUP_ARCHIVELOG_SUMMARY

VIEW_NAME

VIEW_DEFINITION
----
select a.*, case when input_bytes/decode(output_bytes, 0, null,
output_bytes) > 1 then input_bytes/decode(output_bytes, 0, null,
output_bytes) else 1 end compression_ratio,
sys.dbms_rcvman.num2displaysize(input_bytes) input_bytes_display,
sys.dbms_rcvman.num2displaysize(output_bytes) output_bytes_display from (select
sum(num_files_backed) num_files_backed, sum(distinct_files_backed)
distinct_files_backed, min(min_first_change#) min_first_change#,
max(max_next_change#) max_next_change#, min(min_first_time)
min_first_time, max(max_next_time) max_next_time,

VIEW_NAME

VIEW_DEFINITION
----
sum(original_input_bytes) input_bytes, sum(output_bytes) output_bytes
from ((select num_files_backed, distinct_files_backed,
min_first_change#, max_next_change#, min_first_time,
max_next_time, original_input_bytes, output_bytes from (select
count(*) num_files_backed, min(first_change#)min_first_change#,
max(next_change#) max_next_change#, min(first_time)min_first_time,
max(next_time) max_next_time from v$backup_redolog where (set_stamp,
set_count) in (select set_stamp, set_count from
v$backup_set_details)),(select count(*) distinct_files_backed from (select

VIEW_NAME

VIEW_DEFINITION
----
unique thread#, sequence#,resetlogs_change#, resetlogs_time from

v$backup_redolog where (set_stamp, set_count) in (select
set_stamp, set_count from v$backup_set_details))), (select
nvl(sum(original_input_bytes),0) original_input_bytes,
nvl(sum(output_bytes), 0) output_bytes from (select unique
set_count, set_stamp, original_input_bytes, output_bytes
from v$backup_set_details where backup_type='L'))) union (select
num_files_backed, distinct_files_backed, min_first_change#,
max_next_change#, min_first_time, max_next_time,

VIEW_NAME

VIEW_DEFINITION
---
original_input_bytes, output_bytes from (select count(*)
num_files_backed, min(first_change#)min_first_change#,
max(next_change#) max_next_change#, min(first_time)min_first_time,
max(next_time) max_next_time, nvl(sum((blocks+1)*block_size),0)
original_input_bytes, nvl(sum((blocks+1)*block_size),0) output_bytes
from v$proxy_archivedlog a, v$rman_status b, (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionkey skey from dual)c, (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionfromTimeRange fTime from dual) d, (select
/*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime from

VIEW_NAME

VIEW_DEFINITION
---
dual) e where a.status = 'A' and a.rman_status_recid = b.recid
(+) and a.rman_status_stamp = b.stamp (+) and (c.skey is
null or c.skey = b.session_recid) and (d.fTime is null or d.fTime <=
b.start_time) and (e.uTime is null or e.uTime >= b.end_time)),
(select count(*) distinct_files_backed from (select unique thread#,
sequence#, resetlogs_change#,resetlogs_time from v$proxy_archivedlog
a, v$rman_status b, (select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionkey
skey from dual)c, (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionfromTimeRange fTime from dual) d, (select

VIEW_NAME

VIEW_DEFINITION

/*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime from
dual) e where a.status = 'A' and a.rman_status_recid = b.recid
(+) and a.rman_status_stamp = b.stamp (+) and (c.skey is
null or c.skey = b.session_recid) and (d.fTime is null or d.fTime <=
b.start_time) and (e.uTime is null or e.uTime >= b.end_time))))))a

V$BACKUP_ASYNC_IO
select SID, SERIAL, USE_COUNT, RMAN_STATUS_RECID, RMAN_STATUS_STAMP,DEVICE_TYPE,
TYPE, STATUS,FILENAME, SET_COUNT, SET_STAMP, BUFFER_SIZE, BUFFER_COUNT,

VIEW_NAME

```
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
TOTAL_BYTES, OPEN_TIME, CLOSE_TIME, ELAPSED_TIME, MAXOPENFILES, BYTES,
EFFECTIVE_BYTES_PER_SECOND, IO_COUNT, READY, SHORT_WAITS, SHORT_WAIT_TIME_TOTAL,
SHORT_WAIT_TIME_MAX, LONG_WAITS, LONG_WAIT_TIME_TOTAL, LONG_WAIT_TIME_MAX from

gv$backup_async_io where inst_id = userenv('Instance')

V$BACKUP_CONTROLFILE_DETAILS
select a.*,       sys.dbms_rcvman.num2displaysize(filesize) filesize_display
from (select unique 'BACKUPSET' btype, b.recid btype_key,       b.session_recid
session_key,       b.session_recid,       b.session_stamp,       a.set_stamp

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
id1,       b.set_count id2,       creation_time,
resetlogs_change#,resetlogs_time,checkpoint_change#,checkpoint_time,
(datafile_blocks+1)*a.block_size filesize,       1 compression_ratio    from
v$backup_datafile a, v$backup_set_details b where       a.set_stamp =
b.set_stamp and       a.set_count = b.set_count and       file# = 0 union
select unique 'IMAGECOPY' btype, a.recid btype_key,       b.session_recid
session_key,       b.session_recid,       b.session_stamp,       a.recid,
a.stamp,       creation_time,
resetlogs_change#,resetlogs_time,checkpoint_change#,checkpoint_time,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
(blocks+1)*block_size filesize,       1 compression_ratio    from
v$datafile_copy a, v$rman_status b ,   (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionkey skey from dual)c,   (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionfromTimeRange fTime       from dual) d,   (select
/*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime       from
dual) e     where  a.file# = 0 and          a.status = 'A' and
a.rman_status_recid = b.recid (+) and          a.rman_status_stamp = b.stamp
(+) and          (c.skey is null or c.skey = b.session_recid) and
(d.fTime is null or d.fTime <= b.start_time) and          (e.uTime is null or

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
e.uTime >= b.end_time)union select unique 'PROXYCOPY' btype, a.recid btype_key,
b.session_recid session_key,       b.session_recid,       b.session_stamp,
a.recid, a.stamp,       creation_time,
resetlogs_change#,resetlogs_time,checkpoint_change#,checkpoint_time,
(blocks+1)*block_size filesize,       1 compression_ratio    from
v$proxy_datafile a, v$rman_status b,   (select /*+ no_merge */
```

sys.dbms_rcvman.sv_getsessionkey skey from dual)c, (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionfromTimeRange fTime from dual) d, (select
/*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime from

VIEW_NAME

VIEW_DEFINITION
---
dual) e where a.file# = 0 and a.status = 'A' and
a.rman_status_recid = b.recid (+) and a.rman_status_stamp = b.stamp
(+) and (c.skey is null or c.skey = b.session_recid) and
(d.fTime is null or d.fTime <= b.start_time) and (e.uTime is null or
e.uTime >= b.end_time))a

V$BACKUP_CONTROLFILE_SUMMARY
select a.*, case when input_bytes/decode(output_bytes, 0, null,
output_bytes) > 1 then input_bytes/decode(output_bytes, 0, null,

VIEW_NAME

VIEW_DEFINITION
---
output_bytes) else 1 end compression_ratio,
sys.dbms_rcvman.num2displaysize(input_bytes) input_bytes_display,
sys.dbms_rcvman.num2displaysize(output_bytes) output_bytes_display from (select
sum(num_times_backed) num_files_backed, 1 num_distinct_files_backed,
min(min_checkpoint_change#) min_checkpoint_change#,
max(max_checkpoint_change#) max_checkpoint_change#,
min(min_checkpoint_time) min_checkpoint_time, max(max_checkpoint_time)
max_checkpoint_time, sum(input_bytes) input_bytes, sum(output_bytes)
output_bytes from ((select unique count(*) over (partition by

VIEW_NAME

VIEW_DEFINITION
---
creation_time) num_times_backed, min(checkpoint_change#) over
(partition by creation_time) min_checkpoint_change#,
max(checkpoint_change#) over (partition by creation_time)
max_checkpoint_change#, min(checkpoint_time) over (partition by
creation_time) min_checkpoint_time, max(checkpoint_time)
over (partition by creation_time) max_checkpoint_time,
sum((datafile_blocks+1)*block_size) over (partition by
creation_time) input_bytes, sum((blocks+1)*block_size) over (partition by
creation_time) output_bytes, creation_time from

VIEW_NAME

VIEW_DEFINITION
---
v$backup_datafile where file# = 0 and (set_stamp, set_count) in
(select set_stamp, set_count from v$backup_set_details)) union

```
(select unique count(*) over (partition by creation_time)
num_times_backed,      min(checkpoint_change#) over (partition by
creation_time)              min_checkpoint_change#,
max(checkpoint_change#) over (partition by creation_time)
max_checkpoint_change#,      min(checkpoint_time) over (partition by
creation_time)              min_checkpoint_time,      max(checkpoint_time)
over (partition by creation_time)              max_checkpoint_time,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
sum((blocks+1)*block_size) over (partition by creation_time)
input_bytes,       sum((blocks+1)*block_size) over (partition by creation_time)
output_bytes,       creation_time     from v$datafile_copy a, v$rman_status  b,
(select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionkey skey from dual)c,
(select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionfromTimeRange fTime
from dual) d,   (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime        from dual) e     where
a.file# = 0 and a.status = 'A' and          a.rman_status_recid = b.recid (+)
and          a.rman_status_stamp = b.stamp (+) and          (c.skey is null

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
or c.skey = b.session_recid) and          (d.fTime is null or d.fTime <=
b.start_time) and          (e.uTime is null or e.uTime >= b.end_time)     )
union     (select unique count(*) over (partition by creation_time)
num_times_backed,       min(checkpoint_change#) over (partition by
creation_time)              min_checkpoint_change#,
max(checkpoint_change#) over (partition by creation_time)
max_checkpoint_change#,      min(checkpoint_time) over (partition by
creation_time)              min_checkpoint_time,       max(checkpoint_time)
over (partition by creation_time)              max_checkpoint_time,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
sum((blocks+1)*block_size) over (partition by creation_time)
input_bytes,       sum((blocks+1)*block_size) over (partition by creation_time)
output_bytes,       creation_time     from v$proxy_datafile a, v$rman_status
b,   (select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionkey skey from dual)c,
(select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionfromTimeRange fTime
from dual) d,   (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime        from dua

V$BACKUP_COPY_DETAILS

VIEW_NAME
------------------------------
```

```
VIEW_DEFINITION
--------------------------------------------------------------------------------
select a.*,  sys.dbms_rcvman.num2displaysize(output_bytes) output_bytes_display
from (select b.session_recid session_key, b.session_recid, b.session_stamp,
a.recid copy_key, a.file#, a.name, a.tag, a.creation_change#,
a.creation_time, a.checkpoint_change#, a.checkpoint_time,
a.marked_corrupt, (a.blocks+1)*a.block_size output_bytes,
a.completion_time, a.controlfile_type, keep, keep_until,      keep_options,
is_recovery_dest_file  from v$datafile_copy a, v$rman_status b,   (select /*+
no_merge */ sys.dbms_rcvman.sv_getsessionkey skey from dual)c,  (select /*+
no_merge */ sys.dbms_rcvman.sv_getsessionfromTimeRange fTime        from dual)

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
d,   (select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime
from dual) e     where  a.status = 'A' and          a.rman_status_recid =
b.recid (+) and          a.rman_status_stamp = b.stamp (+) and
(c.skey is null or c.skey = b.session_recid) and          (d.fTime is null or
d.fTime <= b.start_time) and          (e.uTime is null or e.uTime >=
b.end_time))a

V$BACKUP_COPY_SUMMARY
select a.*,  sys.dbms_rcvman.num2displaysize(output_bytes) output_bytes_display

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
from (select nvl(sum(num_times_backed),0) num_copies,      sum(distinct_copies)
distinct_copies,      min(min_checkpoint_change#) min_checkpoint_change#,
max(max_checkpoint_change#) max_checkpoint_change#,
min(min_checkpoint_time) min_checkpoint_time,      max(max_checkpoint_time)
max_checkpoint_time,      sum(output_bytes) output_bytes  from     (select
unique file#,count(*) over (partition by file#, creation_change#)
num_times_backed,      count(distinct file#)            over (partition by
file#, creation_change#,checkpoint_change#)           distinct_copies,
min(checkpoint_change#) over (partition by file#, creation_change#)

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
min_checkpoint_change#,      max(checkpoint_change#) over (partition by file#,
creation_change#)              max_checkpoint_change#,
min(checkpoint_time) over (partition by file#, creation_change#)
min_checkpoint_time,      max(checkpoint_time) over (partition by file#,
creation_change#)              max_checkpoint_time,
sum((blocks+1)*block_size) over (partition by file#, creation_change#)
output_bytes    from v$datafile_copy a, v$rman_status  b,  (select /*+
no_merge */ sys.dbms_rcvman.sv_getsessionkey skey from dual)c,  (select /*+
```

no_merge */ sys.dbms_rcvman.sv_getsessionfromTimeRange fTime from dual)

VIEW_NAME

VIEW_DEFINITION
---
d, (select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime
from dual) e where a.status = 'A' and a.rman_status_recid =
b.recid (+) and a.rman_status_stamp = b.stamp (+) and
(c.skey is null or c.skey = b.session_recid) and (d.fTime is null or
d.fTime <= b.start_time) and (e.uTime is null or e.uTime >=
b.end_time)))a

V$BACKUP_CORRUPTION
select RECID , STAMP , SET_STAMP , SET_COUNT , PIECE# , FILE# , BLOCK# , BLOCKS

VIEW_NAME

VIEW_DEFINITION

, CORRUPTION_CHANGE# , MARKED_CORRUPT, CORRUPTION_TYPE from
GV$BACKUP_CORRUPTION
where inst_id = USERENV('Instance')

V$BACKUP_DATAFILE
select RECID , STAMP , SET_STAMP , SET_COUNT , FILE# , CREATION_CHANGE# ,
CREATION_TIME , RESETLOGS_CHANGE# , RESETLOGS_TIME , INCREMENTAL_LEVEL ,
INCREMENTAL_CHANGE# , CHECKPOINT_CHANGE# , CHECKPOINT_TIME ,
ABSOLUTE_FUZZY_CHANGE# , MARKED_CORRUPT , MEDIA_CORRUPT , LOGICALLY_CORRUPT ,
DATAFILE_BLOCKS , BLOCKS , BLOCK_SIZE, OLDEST_OFFLINE_RANGE, COMPLETION_TIME

VIEW_NAME

VIEW_DEFINITION

,CONTROLFILE_TYPE, USED_CHANGE_TRACKING, BLOCKS_READ, USED_OPTIMIZATION,
FOREIGN_DBID, PLUGGED_READONLY, PLUGIN_CHANGE#, PLUGIN_RESETLOGS_CHANGE#,
PLUGIN_RESETLOGS_TIME, SECTION_SIZE, UNDO_OPTIMIZED from GV$BACKUP_DATAFILE
where inst_id = USERENV('Instance')

V$BACKUP_DATAFILE_DETAILS
select a.*, b.ts#, b.name tsname,
sys.dbms_rcvman.num2displaysize(filesize) filesize_display from (select unique
'BACKUPSET' btype, b.recid btype_key, b.session_recid session_key,

VIEW_NAME

VIEW_DEFINITION
---
b.session_recid, b.session_stamp, a.set_stamp id1,
b.set_count id2, file#, creation_change#, creation_time,
resetlogs_change#, resetlogs_time, a.incremental_level,

incremental_change#, checkpoint_change#, checkpoint_time, marked_corrupt,
(datafile_blocks+1)*a.block_size filesize, (datafile_blocks+1)/(blocks+1)
from v$backup_datafile a, v$backup_set_details b where a.set_stamp =
b.set_stamp and a.set_count = b.set_count and file#<>0 union
select unique 'IMAGECOPY' btype, a.recid btype_key, b.session_recid
session_key, b.session_recid, b.session_stamp, a.recid,

VIEW_NAME

VIEW_DEFINITION
----
a.stamp, file#, creation_change#, creation_time,
resetlogs_change#, resetlogs_time, incremental_level, 0
incremental_change#, checkpoint_change#, checkpoint_time, marked_corrupt,
(blocks+1)*block_size filesize, 1 compression_ratio from
v$datafile_copy a, v$rman_status b , (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionkey skey from dual)c, (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionfromTimeRange fTime from dual) d, (select
/*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime from
dual) e where a.file#<>0 and a.status = 'A' and

VIEW_NAME

VIEW_DEFINITION
----
a.rman_status_recid = b.recid (+) and a.rman_status_stamp = b.stamp
(+) and (c.skey is null or c.skey = b.session_recid) and
(d.fTime is null or d.fTime <= b.start_time) and (e.uTime is null or
e.uTime >= b.end_time)union select unique 'PROXYCOPY' btype, a.recid btype_key,
b.session_recid session_key, b.session_recid, b.session_stamp,
a.recid, a.stamp, file#, creation_change#, creation_time,
resetlogs_change#, resetlogs_time, incremental_level, 0
incremental_change#, checkpoint_change#, checkpoint_time, null
marked_corrupt, (blocks+1)*block_size filesize, 1

VIEW_NAME

VIEW_DEFINITION
----
compression_ratio from v$proxy_datafile a, v$rman_status b , (select /*+
no_merge */ sys.dbms_rcvman.sv_getsessionkey skey from dual)c, (select /*+
no_merge */ sys.dbms_rcvman.sv_getsessionfromTimeRange fTime from dual)
d, (select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime
from dual) e where a.file#<>0 and a.status = 'A' and
a.rman_status_recid = b.recid (+) and a.rman_status_stamp = b.stamp
(+) and (c.skey is null or c.skey = b.session_recid) and
(d.fTime is null or d.fTime <= b.start_time) and (e.uTime is null or
e.uTime >= b.end_time)) a, (select df.file#, df.ts#, ts.name from v$datafile df,

VIEW_NAME

VIEW_DEFINITION

```
--------------------------------------------------------------------------------
v$tablespace ts  where ts.ts# = df.ts#) b where a.file# = b.file#(+)

V$BACKUP_DATAFILE_SUMMARY
select a.*,  case when     input_bytes/decode(output_bytes, 0, null,
output_bytes) > 1  then     input_bytes/decode(output_bytes, 0, null,
output_bytes)  else 1 end compression_ratio,
sys.dbms_rcvman.num2displaysize(input_bytes) input_bytes_display,
sys.dbms_rcvman.num2displaysize(output_bytes) output_bytes_display  from (select
sum(num_times_backed) num_files_backed,      count(*)

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
num_distinct_files_backed,     count(distinct ts#)   num_distinct_ts_backed,
min(min_checkpoint_change#) min_checkpoint_change#,
max(max_checkpoint_change#) max_checkpoint_change#,
min(min_checkpoint_time) min_checkpoint_time,     max(max_checkpoint_time)
max_checkpoint_time,     sum(input_bytes) input_bytes,     sum(output_bytes)
output_bytes   from   (select a.*, b.ts#   from   (select unique a.file#,
sum(a.num_times_backed) num_times_backed,        min(min_checkpoint_change#)
min_checkpoint_change#,         max(max_checkpoint_change#)
max_checkpoint_change#,         min(min_checkpoint_time) min_checkpoint_time,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
max(max_checkpoint_time) max_checkpoint_time,         sum(input_bytes)
input_bytes,         sum(output_bytes) output_bytes,
creation_change#    from    ((select unique file#,count(*) over (partition by
file#, creation_change#)         num_times_backed,
min(checkpoint_change#) over (partition by file#, creation_change#)
min_checkpoint_change#,     max(checkpoint_change#) over (partition by file#,
creation_change#)            max_checkpoint_change#,
min(checkpoint_time) over (partition by file#, creation_change#)
min_checkpoint_time,     max(checkpoint_time) over (partition by file#,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
creation_change#)            max_checkpoint_time,
sum((datafile_blocks+1)*block_size)             over (partition by file#,
creation_change#) input_bytes,     sum((blocks+1)*block_size) over (partition
by file#, creation_change#)           output_bytes,     creation_change#
from v$backup_datafile    where file# <> 0 and     (set_stamp, set_count) in
(select set_stamp, set_count from v$backup_set_details)    )    union
(select unique file#, count(*) over (partition by file#,creation_change#)
num_times_backed,     min(checkpoint_change#) over (partition by file#,
creation_change#)            min_checkpoint_change#,
```

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
max(checkpoint_change#) over (partition by file#, creation_change#)
max_checkpoint_change#,      min(checkpoint_time) over (partition by file#,
creation_change#)              min_checkpoint_time,
max(checkpoint_time) over (partition by file#, creation_change#)
max_checkpoint_time,      sum((blocks+1)*block_size) over (partition by file#,
creation_change#)              input_bytes,
sum((blocks+1)*block_size) over (partition by file#, creation_change#)
output_bytes,       creation_change#     from v$datafile_copy a, v$rman_status
b,   (select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionkey skey from dual)c,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
(select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionfromTimeRange fTime
from dual) d,   (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime        from dual) e     where
a.file# <> 0 and a.status = 'A' and          a.rman_status_recid = b.recid (+)
and          a.rman_status_stamp = b.stamp (+) and          (c.skey is null
or c.skey = b.session_recid) and          (d.fTime is null or d.fTime <=
b.start_time) and          (e.uTime is null or e.uTime >= b.end_time)     )
union     (select unique file#, count(*) over (partition by
file#,creation_change#)            num_times_backed,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
min(checkpoint_change#) over (partition by file#, creation_change#)
min_checkpoint_change#,      max(checkpoint_change#) over (partition by file#,
creation_change#)              max_checkpoint_change#,      min(chec

V$BACKUP_DEVICE
select DEVICE_TYPE, DEVICE_NAME FROM GV$BACKUP_DEVICE  where INST_ID =
USERENV('Instance')

V$BACKUP_PIECE

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select  RECID , STAMP ,SET_STAMP , SET_COUNT , PIECE# , COPY#, DEVICE_TYPE ,
HANDLE , COMMENTS , MEDIA , MEDIA_POOL , CONCUR , TAG , STATUS , START_TIME ,
COMPLETION_TIME , ELAPSED_SECONDS, DELETED, BYTES, IS_RECOVERY_DEST_FILE,
RMAN_STATUS_RECID, RMAN_STATUS_STAMP, COMPRESSED, BACKED_BY_VSS, ENCRYPTED,
BACKED_BY_OSB from GV$BACKUP_PIECE where inst_id = USERENV('Instance')
```

V$BACKUP_PIECE_DETAILS
select unique b.session_recid session_key, b.session_recid, b.session_stamp,
a.*, sys.dbms_rcvman.num2displaysize(bytes) size_bytes_display from

VIEW_NAME

VIEW_DEFINITION

(select a.recid bs_key, c.recid bp_key, c.* from v$backup_set a, (select
v$backup_piece.*, count(piece#) over (partition by set_count,
set_stamp, copy#) pieces_per_set from v$backup_piece where status = 'A')
c where a.set_stamp = c.set_stamp and a.set_count = c.set_count and
a.pieces = c.pieces_per_set) a, (select session_recid, session_stamp, recid,
stamp, start_time, end_time from v$rman_status) b , (select /*+ no_merge
/ sys.dbms_rcvman.sv_getsessionkey skey from dual)c, (select /+ no_merge */
sys.dbms_rcvman.sv_getsessionfromTimeRange fTime from dual) d,
(select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime

VIEW_NAME

VIEW_DEFINITION
---
from dual) e where a.rman_status_recid = b.recid (+) and
a.rman_status_stamp = b.stamp (+) and (c.skey is null or c.skey =
b.session_recid) and (d.fTime is null or d.fTime <= b.start_time) and
(e.uTime is null or e.uTime >= b.end_time)

V$BACKUP_REDOLOG
select RECID , STAMP , SET_STAMP , SET_COUNT , THREAD# , SEQUENCE# ,
RESETLOGS_CHANGE# , RESETLOGS_TIME , FIRST_CHANGE# , FIRST_TIME , NEXT_CHANGE# ,
NEXT_TIME , BLOCKS , BLOCK_SIZE, TERMINAL from GV$BACKUP_REDOLOG where inst_id =

VIEW_NAME

VIEW_DEFINITION

USERENV('Instance')

V$BACKUP_SET
select RECID , STAMP , SET_STAMP , SET_COUNT , BACKUP_TYPE ,
CONTROLFILE_INCLUDED , INCREMENTAL_LEVEL , PIECES , START_TIME , COMPLETION_TIME
, ELAPSED_SECONDS , BLOCK_SIZE , INPUT_FILE_SCAN_ONLY, KEEP, KEEP_UNTIL,
KEEP_OPTIONS, MULTI_SECTION from GV$BACKUP_SET where inst_id =
USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$BACKUP_SET_DETAILS

```
select unique b.session_recid session_key, b.session_recid, b.session_stamp,
a.recid bs_key, a.RECID, a.stamp,            a.set_stamp, a.set_count,
a.backup_type, a.controlfile_included,          a.incremental_level,
a.pieces, a.start_time, a.completion_time,          a.elapsed_seconds,
a.block_size, a.keep, a.keep_until,          a.keep_options, a.device_type,
a.compressed,          a.num_copies, a.output_bytes,
a.original_input_bytes, case when a.compression_ratio > 1 then
a.compression_ratio else 1 end, 'A' status,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
a.original_inprate_bytes, a.output_rate_bytes,
sys.dbms_rcvman.num2displaysize(original_input_bytes)
original_input_bytes_display,   sys.dbms_rcvman.num2displaysize(output_bytes)
output_bytes_display ,   sys.dbms_rcvman.num2displaysize(original_inprate_bytes)
original_inprate_bytes_display,
sys.dbms_rcvman.num2displaysize(output_rate_bytes)
output_rate_bytes_display,   sys.dbms_rcvman.sec2displaytime(elapsed_seconds)
time_taken_display,    a.encrypted, a.backed_by_osb    from    (   select unique
a.*,      b.rman_status_recid, b.rman_status_stamp,      decode(b.devcnt, 1,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
first_value(b.device_type) over                              (partition
by b.set_stamp, b.set_count),                        '*') device_type,
b.compressed,       count(distinct copy#)           over (partition by
b.set_stamp, b.set_count) num_copies,       b.output_bytes output_bytes,
c.original_input_bytes,       c.original_input_bytes /
(decode(b.output_bytes,0,c.original_input_bytes,b.output_bytes))
compression_ratio,       c.original_input_bytes/
(decode(a.elapsed_seconds, 0, 1, a.elapsed_seconds))
original_inprate_bytes,       b.output_bytes/

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
(decode(a.elapsed_seconds, 0, 1, a.elapsed_seconds))
output_rate_bytes,       b.encrypted, b.backed_by_osb    from v$backup_set a,
(select set_stamp, set_count, device_type, status,   count(distinct device_type)
over (partition by set_count,set_stamp)devcnt,    compressed, encrypted,
backed_by_osb,    sum(bytes) over (partition by set_count, set_stamp, copy#)
output_bytes,    copy#, RMAN_STATUS_RECID, RMAN_STATUS_STAMP,    count(piece#)
over (partition by set_count, set_stamp, copy#) npieces    from v$backup_piece
where status = 'A') b,   (    select set_stamp, set_count,
sum(original_input_bytes)          original_input_bytes from    (      select

VIEW_NAME
```

```
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
set_stamp, set_count,       sum((datafile_blocks+1)*block_size)         over
(partition by set_count, set_stamp) original_input_bytes       from
v$backup_datafile       union       select set_stamp, set_count,
sum(bytes)         over (partition by set_count, set_stamp)
original_input_bytes       from v$backup_spfile     ) group by set_stamp,
set_count     union     select set_stamp, set_count,
sum((blocks+1)*block_size)        over (partition by set_count, set_stamp)
original_input_bytes     from v$backup_redolog     ) c     where
a.set_stamp=b.set_stamp and a.set_stamp=b.set_stamp and

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
a.set_stamp=c.set_stamp and a.set_stamp=c.set_stamp and
a.pieces=b.npieces     ) a,     (select session_recid, session_stamp, recid,
stamp, start_time, end_time     from v$rman_status) b,     (select /*+ no_merge
*/ sys.dbms_rcvman.sv_getsessionkey skey from dual)c,   (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionfromTimeRange fTime         from dual) d,
(select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime
from dual) e     where  a.rman_status_recid = b.recid (+) and
a.rman_status_stamp = b.stamp (+) and           (c.skey is null or c.skey =
b.session_recid) and           (d.fTime is null or d.fTime <= b.start_time) and

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
(e.uTime is null or e.uTime >= b.end_time)

V$BACKUP_SET_SUMMARY
select a.*,   case when      original_input_bytes/decode(output_bytes, 0, null,
output_bytes) > 1   then      original_input_bytes/decode(output_bytes, 0, null,
output_bytes)   else 1 end compression_ratio,
sys.dbms_rcvman.num2displaysize(original_input_bytes)
original_input_bytes_display,   sys.dbms_rcvman.num2displaysize(output_bytes)
output_bytes_display ,   sys.dbms_rcvman.num2displaysize(original_inprate_bytes)

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
original_inprate_bytes_display,
sys.dbms_rcvman.num2displaysize(output_rate_bytes)
output_rate_bytes_display from (select count(*) num_backupsets,
min(start_time) oldest_backup_time,       max(start_time) newest_backup_time,
sum(output_bytes) output_bytes,       sum(original_input_bytes)
original_input_bytes,       avg(original_inprate_bytes) original_inprate_bytes,
avg(output_rate_bytes) output_rate_bytes from   (select unique      set_stamp,
```

set_count, start_time, output_bytes, original_input_bytes,
original_inprate_bytes, output_rate_bytes, compression_ratio from

VIEW_NAME

VIEW_DEFINITION
----
v$backup_set_details))a

V$BACKUP_SPFILE
select RECID , STAMP , SET_STAMP , SET_COUNT , MODIFICATION_TIME, BYTES ,
COMPLETION_TIME, DB_UNIQUE_NAME from GV$BACKUP_SPFILE where inst_id =
USERENV('Instance')

V$BACKUP_SPFILE_DETAILS
select unique b.session_recid session_key, b.session_recid,

VIEW_NAME

VIEW_DEFINITION
----
b.session_stamp, b.recid bs_key, a.set_stamp, b.set_count,
modification_time, a.bytes,
sys.dbms_rcvman.num2displaysize(a.bytes) filesize_display from
v$backup_spfile a, v$backup_set_details b where a.set_stamp =
b.set_stamp and a.set_count = b.set_count

V$BACKUP_SPFILE_SUMMARY
select num_files_backed, num_distinct_files_backed,
min_modification_time, max_modification_time, input_bytes,

VIEW_NAME

VIEW_DEFINITION
----
sys.dbms_rcvman.num2displaysize(input_bytes) input_bytes_display from (select
count(*) num_files_backed, min(modification_time)min_modification_time,
max(modification_time) max_modification_time, sum(bytes) input_bytes
from v$backup_spfile where (set_stamp, set_count) in (select set_stamp,
set_count from v$backup_set_details)), (select count(*)
num_distinct_files_backed from (select unique modification_time
from v$backup_spfile where (set_stamp, set_count) in
(select set_stamp, set_count from v$backup_set_details)))

VIEW_NAME

VIEW_DEFINITION

V$BACKUP_SYNC_IO
select SID, SERIAL, USE_COUNT, RMAN_STATUS_RECID, RMAN_STATUS_STAMP,DEVICE_TYPE,
TYPE, STATUS, FILENAME, SET_COUNT, SET_STAMP, BUFFER_SIZE, BUFFER_COUNT,

TOTAL_BYTES, OPEN_TIME, CLOSE_TIME, ELAPSED_TIME, MAXOPENFILES, BYTES,
EFFECTIVE_BYTES_PER_SECOND, IO_COUNT, IO_TIME_TOTAL, IO_TIME_MAX,
DISCRETE_BYTES_PER_SECOND from gv$backup_sync_io where inst_id =
userenv('Instance')

V$BGPROCESS

VIEW_NAME

VIEW_DEFINITION
----
select paddr,pserial#,name,description,error from gv$bgprocess where inst_id =
USERENV('Instance')

V$BH
select file#, block#, class#, status, xnc, forced_reads, forced_writes,
lock_element_addr, lock_element_name, lock_element_class, dirty, temp, ping,
stale, direct, new, objd, ts# from gv$bh where inst_id = USERENV('Instance')

V$BLOCKING_QUIESCE

VIEW_NAME

VIEW_DEFINITION
----
select sid from gv$blocking_quiesce   where inst_id = userenv('instance')

V$BLOCK_CHANGE_TRACKING
select decode(di2ctst, 0, 'DISABLED', 1, 'TRANSITION', 2,
'ENABLED','ERROR'),fnnam, fh.bytes from x$kccdi2, x$kccfn, (select
max(fhfsz*fhbsz) bytes from x$krcfh) fh where fnnum(+)=di2ctfn and fntyp(+)=200

V$BSP
select cr_requests, current_requests, data_requests,   undo_requests,

VIEW_NAME

VIEW_DEFINITION
----
tx_requests, current_results,   private_results, zero_results,
disk_read_results,   fail_results, fairness_down_converts,
fairness_clears,   free_lock_elements, flushes, flushes_queued,
flush_queue_full, flush_max_time, light_works, errors   from gv$bsp
where inst_id = USERENV('Instance')

V$BUFFERED_PUBLISHERS
select queue_id, queue_schema, queue_name, sender_name, sender_address,
sender_protocol, num_msgs, cnum_msgs, last_enqueued_msg, unbrowsed_msgs,

VIEW_NAME

VIEW_DEFINITION

```
--------------------------------------------------------------------------------
overspilled_msgs, memory_usage, publisher_state  from gv$buffered_publishers
where inst_id = USERENV('Instance')

V$BUFFERED_QUEUES
select queue_id, queue_schema, queue_name, startup_time,  num_msgs, spill_msgs,
cnum_msgs, cspill_msgs, expired_msgs from gv$buffered_queues where inst_id =
USERENV('Instance')

V$BUFFERED_SUBSCRIBERS

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select queue_id, queue_schema, queue_name, subscriber_id, subscriber_name,
subscriber_address, protocol, subscriber_type,  startup_time, last_browsed_seq,
last_browsed_num, last_dequeued_seq, last_dequeued_num, current_enq_seq,
num_msgs, cnum_msgs, total_dequeued_msg, total_spilled_msg, expired_msgs,
message_lag  from gv$buffered_subscribers where inst_id = USERENV('Instance')

V$BUFFER_POOL
select id, name, block_size, resize_state, current_size, buffers,
target_size, target_buffers, prev_size, prev_buffers,               lo_bnum,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
hi_bnum, lo_setid, hi_setid, set_count             from gv$buffer_pool  where
inst_id = USERENV('Instance')

V$BUFFER_POOL_STATISTICS
select id, name, block_size, set_msize, cnum_repl, cnum_write, cnum_set,
buf_got, sum_write, sum_scan, free_buffer_wait, write_complete_wait,
buffer_busy_wait, free_buffer_inspected, dirty_buffers_inspected,
db_block_change, db_block_gets, consistent_gets, physical_reads, physical_writes
from gv$buffer_pool_statistics where inst_id = USERENV('Instance')

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

V$CALLTAG
select session_id, performance_class, work_request_class,        hop_count,
service_name, module, action, username, program        from gv$calltag where
inst_id = USERENV('Instance')

V$CIRCUIT
select  CIRCUIT , DISPATCHER , SERVER , WAITER , SADDR , STATUS , QUEUE ,
MESSAGE0 , MESSAGE1 , MESSAGE2, MESSAGE3, MESSAGES , BYTES , BREAKS ,
```

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
PRESENTATION, PCIRCUIT from GV$CIRCUIT where inst_id = USERENV('Instance')

V$CLASS_CACHE_TRANSFER
select class,                                   x_2_null,
x_2_null_forced_write,          x_2_null_forced_stale, x_2_s,
x_2_s_forced_write,    s_2_null,                    s_2_null_forced_stale,
null_2_x,                   s_2_x,              null_2_s,
cr_transfer,          current_transfer              from
gv$class_cache_transfer                       where inst_id =

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
USERENV('Instance')

V$CLASS_PING
select class, x_2_null, x_2_null_forced_write, x_2_null_forced_stale,  x_2_s,
x_2_s_forced_write, x_2_ssx, x_2_ssx_forced_write, s_2_null,
s_2_null_forced_stale, ss_2_null, ss_2_rls, op_2_ss, null_2_x, s_2_x,  ssx_2_x,
null_2_s, null_2_ss from gv$class_ping                      where inst_id =
USERENV('Instance')


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
V$CLIENT_RESULT_CACHE_STATS
select  CLIENT_REGID,       BLOCK_SIZE,       BLOCK_MAX,       BLOCK_COUNT,
BUCKET_COUNT,       CREATE_SUCC,       CREATE_FAIL,       FINDS,
INVALIDATIONS,       DELETE_INVALIDS,       DELETE_VALIDS from
GV$CLIENT_RESULT_CACHE_STATS where inst_id=USERENV('Instance')

V$CLIENT_STATS
select  CLIENT_IDENTIFIER , STAT_ID , STAT_NAME ,  VALUE from GV$CLIENT_STATS
where inst_id = USERENV('Instance')

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

V$CLUSTER_INTERCONNECTS
SELECT NAME, IP_ADDRESS, IS_PUBLIC, SOURCE         FROM
GV$CLUSTER_INTERCONNECTS  WHERE         INST_ID=USERENV('Instance')
```

V$CONFIGURED_INTERCONNECTS
SELECT NAME, IP_ADDRESS, IS_PUBLIC, SOURCE FROM
GV$CONFIGURED_INTERCONNECTS WHERE INST_ID=USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$CONTEXT
select namespace, attribute, value from x$context

V$CONTROLFILE
select STATUS , NAME, IS_RECOVERY_DEST_FILE, BLOCK_SIZE, FILE_SIZE_BLKS from
GV$CONTROLFILE where inst_id = USERENV('Instance')

V$CONTROLFILE_RECORD_SECTION
select TYPE , RECORD_SIZE , RECORDS_TOTAL , RECORDS_USED , FIRST_INDEX ,

VIEW_NAME

VIEW_DEFINITION

LAST_INDEX , LAST_RECID from GV$CONTROLFILE_RECORD_SECTION where inst_id =
USERENV('Instance')

V$COPY_CORRUPTION
select RECID , STAMP , COPY_RECID , COPY_STAMP , FILE# , BLOCK# , BLOCKS ,
CORRUPTION_CHANGE# , MARKED_CORRUPT, CORRUPTION_TYPE from GV$COPY_CORRUPTION

where inst_id = USERENV('Instance')

V$CORRUPT_XID_LIST

VIEW_NAME

VIEW_DEFINITION
---
select CORRUPT_XID from GV$CORRUPT_XID_LIST where inst_id=USERENV('instance')

V$CPOOL_CC_INFO
select POOL_NAME, CCLASS_NAME from GV$CPOOL_CC_INFO where
inst_id=USERENV('Instance')

V$CPOOL_CC_STATS
select CCLASS_NAME, NUM_REQUESTS, NUM_HITS, NUM_MISSES,
NUM_WAITS, WAIT_TIME, CLIENT_REQ_TIMEOUTS,

VIEW_NAME

VIEW_DEFINITION

NUM_AUTHENTICATIONS from GV$CPOOL_CC_STATS where inst_id=USERENV('Instance')

V$CPOOL_STATS
select POOL_NAME, NUM_OPEN_SERVERS, NUM_BUSY_SERVERS,
NUM_AUTH_SERVERS, NUM_REQUESTS, NUM_HITS, NUM_MISSES,
NUM_WAITS, WAIT_TIME, CLIENT_REQ_TIMEOUTS,
NUM_AUTHENTICATIONS, NUM_PURGED, HISTORIC_MAX from GV$CPOOL_STATS
where inst_id=USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$CR_BLOCK_SERVER
select cr_requests, current_requests, data_requests,
undo_requests, tx_requests, other_requests, current_results,
private_results, zero_results, disk_read_results,
fail_results, stale, fairness_down_converts, fairness_clears,
free_gc_elements, flushes, flushes_queued,
flush_queue_full, flush_max_time, light_works, errors from
gv$cr_block_server where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$CURRENT_BLOCK_SERVER
select pin1, pin10, pin100, pin1000, pin10000, flush1, flush10,
flush100, flush1000, flush10000, write1, write10, write100, write1000,
write10000, cleandc, rcvdc, queuedc, evictdc, writedc
from gv$current_block_server where inst_id = USERENV('Instance')

V$DATABASE
select DBID, NAME, CREATED, RESETLOGS_CHANGE#, RESETLOGS_TIME,
PRIOR_RESETLOGS_CHANGE#, PRIOR_RESETLOGS_TIME,LOG_MODE, CHECKPOINT_CHANGE#,

VIEW_NAME

VIEW_DEFINITION

ARCHIVE_CHANGE#, CONTROLFILE_TYPE, CONTROLFILE_CREATED, CONTROLFILE_SEQUENCE#,
CONTROLFILE_CHANGE#, CONTROLFILE_TIME, OPEN_RESETLOGS, VERSION_TIME, OPEN_MODE,
PROTECTION_MODE, PROTECTION_LEVEL, REMOTE_ARCHIVE, ACTIVATION#, SWITCHOVER#,
DATABASE_ROLE, ARCHIVELOG_CHANGE#, ARCHIVELOG_COMPRESSION, SWITCHOVER_STATUS,

DATAGUARD_BROKER, GUARD_STATUS, SUPPLEMENTAL_LOG_DATA_MIN,
SUPPLEMENTAL_LOG_DATA_PK, SUPPLEMENTAL_LOG_DATA_UI, FORCE_LOGGING, PLATFORM_ID,

PLATFORM_NAME, RECOVERY_TARGET_INCARNATION#, LAST_OPEN_INCARNATION#,

CURRENT_SCN, FLASHBACK_ON,SUPPLEMENTAL_LOG_DATA_FK, SUPPLEMENTAL_LOG_DATA_ALL,

DB_UNIQUE_NAME, STANDBY_BECAME_PRIMARY_SCN, FS_FAILOVER_STATUS,

VIEW_NAME

VIEW_DEFINITION

FS_FAILOVER_CURRENT_TARGET, FS_FAILOVER_THRESHOLD,
FS_FAILOVER_OBSERVER_PRESENT,
FS_FAILOVER_OBSERVER_HOST, CONTROLFILE_CONVERTED, PRIMARY_DB_UNIQUE_NAME,
SUPPLEMENTAL_LOG_DATA_PL, MIN_REQUIRED_CAPTURE_CHANGE# from GV$DATABASE where

inst_id = USERENV('Instance')

V$DATABASE_BLOCK_CORRUPTION
select FILE#, BLOCK#, BLOCKS, CORRUPTION_CHANGE#, CORRUPTION_TYPE from
GV$DATABASE_BLOCK_CORRUPTION where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$DATABASE_INCARNATION
select incarnation#, resetlogs_change#, resetlogs_time,
prior_resetlogs_change#, prior_resetlogs_time, status, resetlogs_id,
prior_incarnation#, flashback_database_allowed from GV$DATABASE_INCARNATION
where inst_id = USERENV('Instance')

V$DATAFILE
select FILE# , CREATION_CHANGE# , CREATION_TIME , TS# , RFILE# , STATUS ,
ENABLED , CHECKPOINT_CHANGE# , CHECKPOINT_TIME, UNRECOVERABLE_CHANGE#,

VIEW_NAME

VIEW_DEFINITION

UNRECOVERABLE_TIME, LAST_CHANGE# , LAST_TIME , OFFLINE_CHANGE# , ONLINE_CHANGE#
, ONLINE_TIME , BYTES , BLOCKS , CREATE_BYTES , BLOCK_SIZE , NAME, PLUGGED_IN,
BLOCK1_OFFSET , AUX_NAME , FIRST_NONLOGGED_SCN, FIRST_NONLOGGED_TIME,
FOREIGN_DBID, FOREIGN_CREATION_CHANGE#, FOREIGN_CREATION_TIME,
PLUGGED_READONLY,
PLUGIN_CHANGE#, PLUGIN_RESETLOGS_CHANGE#, PLUGIN_RESETLOGS_TIME from
GV$DATAFILE
where inst_id = USERENV('Instance')

V$DATAFILE_COPY
select RECID , STAMP , NAME , TAG , FILE# , RFILE# , CREATION_CHANGE# ,

VIEW_NAME

VIEW_DEFINITION

CREATION_TIME , RESETLOGS_CHANGE# , RESETLOGS_TIME , INCREMENTAL_LEVEL,
CHECKPOINT_CHANGE# , CHECKPOINT_TIME , ABSOLUTE_FUZZY_CHANGE# ,
RECOVERY_FUZZY_CHANGE# , RECOVERY_FUZZY_TIME , ONLINE_FUZZY , BACKUP_FUZZY ,
MARKED_CORRUPT , MEDIA_CORRUPT , LOGICALLY_CORRUPT , BLOCKS , BLOCK_SIZE ,
OLDEST_OFFLINE_RANGE, DELETED, STATUS, COMPLETION_TIME , CONTROLFILE_TYPE, KEEP,
KEEP_UNTIL, KEEP_OPTIONS, SCANNED, IS_RECOVERY_DEST_FILE, RMAN_STATUS_RECID,
RMAN_STATUS_STAMP, CONVERTED_FILE, SAME_ENDIAN, FOREIGN_DBID, PLUGGED_READONLY,

PLUGIN_CHANGE#, PLUGIN_RESETLOGS_CHANGE#, PLUGIN_RESETLOGS_TIME, BACKED_BY_VSS

from GV$DATAFILE_COPY where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$DATAFILE_HEADER
select FILE# , STATUS , ERROR , FORMAT, RECOVER , FUZZY, CREATION_CHANGE#,
CREATION_TIME , TABLESPACE_NAME , TS# , RFILE# , RESETLOGS_CHANGE# ,
RESETLOGS_TIME , CHECKPOINT_CHANGE# , CHECKPOINT_TIME , CHECKPOINT_COUNT , BYTES
, BLOCKS , NAME, SPACE_HEADER, LAST_DEALLOC_CHANGE#, UNDO_OPT_CURRENT_CHANGE#
from GV$DATAFILE_HEADER where inst_id = USERENV('Instance')

V$DATAGUARD_CONFIG

VIEW_NAME

VIEW_DEFINITION
----
select DGCDBUN from x$krstdgc

V$DATAGUARD_STATS
select NAME, VALUE, UNIT, TIME_COMPUTED FROM x$krvslvas union select NAME,
VALUE, UNIT, TIME_COMPUTED from x$krstpvrs

V$DATAGUARD_STATUS
select FACILITY, SEVERITY, DEST_ID, MESSAGE_NUM, ERROR_CODE, CALLOUT, TIMESTAMP,
MESSAGE from GV$DATAGUARD_STATUS where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$DATAPUMP_JOB
SELECT job_id, job_name, owner_name, msg_ctrl_queue,
status_queue, operation, job_mode, master_id, state, workers,
flags, serialnum FROM gv$datapump_job WHERE inst_id =
USERENV('INSTANCE')

V$DATAPUMP_SESSION
SELECT attach_id, job_id, saddr, type FROM gv$datapump_session

VIEW_NAME

VIEW_DEFINITION

WHERE inst_id = USERENV('INSTANCE')

V$DBFILE
select FILE# , NAME from GV$DBFILE where inst_id = USERENV('Instance')

V$DBLINK
select DB_LINK , OWNER_ID , LOGGED_ON , HETEROGENEOUS , PROTOCOL , OPEN_CURSORS , IN_TRANSACTION , UPDATE_SENT , COMMIT_POINT_STRENGTH from GV$DBLINK where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$DB_CACHE_ADVICE
select id, name, block_size, advice_status, size_for_estimate, size_factor, buffers_for_estimate, estd_physical_read_factor, estd_physical_reads, estd_physical_read_time, estd_pct_of_db_time_for_reads, estd_cluster_reads, estd_cluster_read_time from gv$db_cache_advice where inst_id = userenv('instance')

VIEW_NAME

VIEW_DEFINITION

V$DB_OBJECT_CACHE
select OWNER , NAME , DB_LINK , NAMESPACE , TYPE , SHARABLE_MEM , LOADS , EXECUTIONS , LOCKS , PINS , KEPT , CHILD_LATCH , INVALIDATIONS from GV$DB_OBJECT_CACHE where inst_id = USERENV('Instance')

V$DB_PIPES
select OWNERID , NAME , TYPE , PIPE_SIZE from GV$DB_PIPES where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$DB_TRANSPORTABLE_PLATFORM
select PLATFORM_ID, PLATFORM_NAME, decode(endian_format, 1,'Big'

```
,0,'Little','UNKNOWN FORMAT')         from x$kcpxpl         where
endian_format =          (select endian_format from x$kcpxpl pl, x$kccdi di
where pl.platform_id = di.diplid)
```

V$DELETED_OBJECT

```
select  RECID, STAMP, TYPE, OBJECT_RECID, OBJECT_STAMP, OBJECT_DATA,  SET_STAMP,
SET_COUNT from GV$DELETED_OBJECT where inst_id = USERENV('Instance')
```

VIEW_NAME

VIEW_DEFINITION

V$DETACHED_SESSION

```
SELECT INDX, PG_NAME, SID, SERIAL#, PID    FROM gv$detached_session WHERE
INST_ID = USERENV('INSTANCE')
```

V$DIAG_INFO

```
SELECT inst_id, name, value FROM gv$diag_info         WHERE  inst_id =
USERENV('INSTANCE')
```

VIEW_NAME

VIEW_DEFINITION

V$DISPATCHER

```
select  NAME , NETWORK , PADDR , STATUS , ACCEPT , MESSAGES , BYTES , BREAKS ,
OWNED , CREATED , IDLE , BUSY , LISTENER, CONF_INDX from GV$DISPATCHER where
inst_id = USERENV('Instance')
```

V$DISPATCHER_CONFIG

```
select conf_indx, network, dispatchers, connections, sessions,  pool, ticks,
inbd_timout, outbd_timout, multiplex, listener, service from
GV$DISPATCHER_CONFIG where inst_id = USERENV('Instance')
```

VIEW_NAME

VIEW_DEFINITION

V$DISPATCHER_RATE

```
select  NAME,
PADDR,CUR_LOOP_RATE,CUR_EVENT_RATE,CUR_EVENTS_PER_LOOP,CUR_MSG_RATE,CUR_SVR_B
UF_
RATE,CUR_SVR_BYTE_RATE,CUR_SVR_BYTE_PER_BUF,CUR_CLT_BUF_RATE,CUR_CLT_BYTE_RATE,
C
UR_CLT_BYTE_PER_BUF,CUR_BUF_RATE,CUR_BYTE_RATE,CUR_BYTE_PER_BUF,CUR_IN_CONNECT
_R
ATE,CUR_OUT_CONNECT_RATE,CUR_RECONNECT_RATE,MAX_LOOP_RATE,MAX_EVENT_RATE,MAX
_EVE
NTS_PER_LOOP,MAX_MSG_RATE,MAX_SVR_BUF_RATE,MAX_SVR_BYTE_RATE,MAX_SVR_BYTE_PE
```

R_BU
F,MAX_CLT_BUF_RATE,MAX_CLT_BYTE_RATE,MAX_CLT_BYTE_PER_BUF,MAX_BUF_RATE,MAX_BYT
E_

VIEW_NAME

VIEW_DEFINITION

RATE,MAX_BYTE_PER_BUF,MAX_IN_CONNECT_RATE,MAX_OUT_CONNECT_RATE,MAX_RECONNECT
_RAT
E,AVG_LOOP_RATE,AVG_EVENT_RATE,AVG_EVENTS_PER_LOOP,AVG_MSG_RATE,AVG_SVR_BUF_R
ATE
,AVG_SVR_BYTE_RATE,AVG_SVR_BYTE_PER_BUF,AVG_CLT_BUF_RATE,AVG_CLT_BYTE_RATE,AVG
_C
LT_BYTE_PER_BUF,AVG_BUF_RATE,AVG_BYTE_RATE,AVG_BYTE_PER_BUF,AVG_IN_CONNECT_RAT
E,
AVG_OUT_CONNECT_RATE,AVG_RECONNECT_RATE,TTL_LOOPS,TTL_MSG,TTL_SVR_BUF,TTL_CLT_
BU
F,TTL_BUF,TTL_IN_CONNECT,TTL_OUT_CONNECT,TTL_RECONNECT,SCALE_LOOPS,SCALE_MSG,SC
A
LE_SVR_BUF,SCALE_CLT_BUF,SCALE_BUF,SCALE_IN_CONNECT,SCALE_OUT_CONNECT,SCALE_RE
CO
NNECT from GV$DISPATCHER_RATE where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$DLM_ALL_LOCKS
select LOCKP, GRANT_LEVEL, REQUEST_LEVEL, RESOURCE_NAME1,
RESOURCE_NAME2, PID, TRANSACTION_ID0, TRANSACTION_ID1, GROUP_ID,
OPEN_OPT_DEADLOCK, OPEN_OPT_PERSISTENT, OPEN_OPT_PROCESS_OWNED,
OPEN_OPT_NO_XID, CONVERT_OPT_GETVALUE, CONVERT_OPT_PUTVALUE,
CONVERT_OPT_NOVALUE, CONVERT_OPT_DUBVALUE, CONVERT_OPT_NOQUEUE,
CONVERT_OPT_EXPRESS, CONVERT_OPT_NODEADLOCKWAIT,
CONVERT_OPT_NODEADLOCKBLOCK, WHICH_QUEUE, LOCKSTATE, AST_EVENT0,
OWNER_NODE, BLOCKED, BLOCKER from GV$DLM_ALL_LOCKS where INST_ID =

VIEW_NAME

VIEW_DEFINITION

USERENV('Instance')

V$DLM_CONVERT_LOCAL
select INST_ID, CONVERT_TYPE, AVERAGE_CONVERT_TIME, CONVERT_COUNT from
GV$DLM_CONVERT_LOCAL where INST_ID = USERENV('Instance')

V$DLM_CONVERT_REMOTE
select INST_ID, CONVERT_TYPE, AVERAGE_CONVERT_TIME, CONVERT_COUNT from
GV$DLM_CONVERT_REMOTE where INST_ID = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$DLM_LATCH
select addr, latch#, level#, name, gets, misses, sleeps,immediate_gets,
immediate_misses, waiters_woken, waits_holding_latch, spin_gets, sleep1,
sleep2, sleep3, sleep4, sleep5, sleep6, sleep7, sleep8, sleep9, sleep10,
sleep11, wait_time from GV$DLM_LATCH where INST_ID = USERENV('Instance')

V$DLM_LOCKS
select LOCKP, GRANT_LEVEL, REQUEST_LEVEL, RESOURCE_NAME1,

VIEW_NAME

VIEW_DEFINITION

RESOURCE_NAME2, PID, TRANSACTION_ID0, TRANSACTION_ID1, GROUP_ID,
OPEN_OPT_DEADLOCK, OPEN_OPT_PERSISTENT, OPEN_OPT_PROCESS_OWNED,
OPEN_OPT_NO_XID, CONVERT_OPT_GETVALUE, CONVERT_OPT_PUTVALUE,
CONVERT_OPT_NOVALUE, CONVERT_OPT_DUBVALUE, CONVERT_OPT_NOQUEUE,
CONVERT_OPT_EXPRESS, CONVERT_OPT_NODEADLOCKWAIT,
CONVERT_OPT_NODEADLOCKBLOCK, WHICH_QUEUE, LOCKSTATE, AST_EVENT0,
OWNER_NODE, BLOCKED, BLOCKER from GV$DLM_LOCKS where INST_ID =
USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$DLM_MISC
select STATISTIC#, NAME, VALUE FROM GV$DLM_MISC where INST_ID =
USERENV('Instance')

V$DLM_RESS
select RESP, RESOURCE_NAME, ON_CONVERT_Q, ON_GRANT_Q, PERSISTENT_RES,
MASTER_NODE, NEXT_CVT_LEVEL, VALUE_BLK_STATE, VALUE_BLK from
GV$DLM_RESS where INST_ID = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$DLM_TRAFFIC_CONTROLLER
select LOCAL_NID,REMOTE_NID,REMOTE_RID,REMOTE_INC,
TCKT_AVAIL,TCKT_LIMIT,TCKT_RCVD,TCKT_WAIT, SND_SEQ_NO,RCV_SEQ_NO,
SND_Q_LEN, SND_Q_MAX, SND_Q_TOT, SND_Q_TM_BASE, SND_Q_TM_WRAP,
STATUS,SND_PROXY from GV$DLM_TRAFFIC_CONTROLLER where INST_ID

```
= userenv('instance')

V$DNFS_CHANNELS
select pnum, svrname, path, ch_id, svr_id, sends,         recvs, pings

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
from gv$dnfs_channels where inst_id = USERENV('Instance')

V$DNFS_FILES
select filename, filesize, pnum, svr_id         from gv$dnfs_files where
inst_id = USERENV('Instance')

V$DNFS_SERVERS
select id, svrname, dirname, mntport, nfsport, wtmax,         rtmax from
gv$dnfs_servers where inst_id = USERENV('Instance')

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

V$DNFS_STATS
select pnum, nfs_null, nfs_getattr, nfs_setattr,         nfs_lookup,
nfs_access, nfs_readlink, nfs_read, nfs_write,         nfs_create, nfs_mkdir,
nfs_symlink, nfs_mknod, nfs_remove, nfs_rmdir,         nfs_rename, nfs_link,
nfs_readdir, nfs_readdirplus,         nfs_fsstat, nfs_fsinfo, nfs_pathconf,
nfs_commit, nfs_mount         from gv$dnfs_stats where inst_id =
USERENV('Instance')


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
V$DYNAMIC_REMASTER_STATS
select REMASTER_OPS, REMASTER_TIME, REMASTERED_OBJECTS, QUIESCE_TIME,
FREEZE_TIME, CLEANUP_TIME, REPLAY_TIME, FIXWRITE_TIME, SYNC_TIME,
RESOURCES_CLEANED, REPLAYED_LOCKS_SENT, REPLAYED_LOCKS_RECEIVED,
CURRENT_OBJECTS FROM GV$DYNAMIC_REMASTER_STATS  where INST_ID =
USERENV('Instance')

V$ENABLEDPRIVS
select  PRIV_NUMBER from GV$ENABLEDPRIVS where inst_id = USERENV('Instance')

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
```

V$ENCRYPTED_TABLESPACES
select TS#, ENCRYPTIONALG, ENCRYPTEDTS from
GV$ENCRYPTED_TABLESPACES where INST_ID =
USERENV('Instance')

V$ENCRYPTION_WALLET
SELECT WRL_TYPE, WRL_PARAMETER, STATUS FROM GV$ENCRYPTION_WALLET WHERE INST_ID
=
USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$ENQUEUE_LOCK
select ADDR , KADDR , SID , TYPE , ID1 , ID2 , LMODE , REQUEST , CTIME , BLOCK
from GV$ENQUEUE_LOCK where inst_id = USERENV('Instance')

V$ENQUEUE_STAT
select INST_ID, EQ_TYPE, TOTAL_REQ#, TOTAL_WAIT#, SUCC_REQ#, FAILED_REQ#,
CUM_WAIT_TIME from GV$ENQUEUE_STAT where INST_ID = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$ENQUEUE_STATISTICS
select EQ_NAME, EQ_TYPE, REQ_REASON, TOTAL_REQ#, TOTAL_WAIT#,
SUCC_REQ#, FAILED_REQ#, CUM_WAIT_TIME, REQ_DESCRIPTION, EVENT# from
GV$ENQUEUE_STATISTICS where INST_ID = USERENV('Instance')

V$EVENTMETRIC
SELECT begin_time, end_time, intsize_csec, event#, event_id,
num_sess_waiting, time_waited, wait_count FROM gv$eventmetric
WHERE inst_id = USERENV('INSTANCE')

VIEW_NAME

VIEW_DEFINITION

V$EVENT_HISTOGRAM
select event#, event, wait_time_milli, wait_count from gv$event_histogram where
inst_id = USERENV('Instance')

V$EVENT_NAME
select event#, event_id, name,parameter1,parameter2,parameter3, wait_class_id,
wait_class#, wait_class from gv$event_name where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$EXECUTION
select PID , DEPTH , FUNCTION , TYPE , NVALS , VAL1 , VAL2 , SEQH , SEQL from GV$EXECUTION where inst_id = USERENV('Instance')

V$FAST_START_SERVERS
SELECT STATE, UNDOBLOCKSDONE, PID, XID from GV$FAST_START_SERVERS where inst_id = USERENV('Instance')

V$FAST_START_TRANSACTIONS

VIEW_NAME

VIEW_DEFINITION

SELECT USN, SLT, SEQ, STATE, UNDOBLOCKSDONE, UNDOBLOCKSTOTAL, PID, CPUTIME, PARENTUSN, PARENTSLT, PARENTSEQ, XID, PXID, RCVSERVERS from GV$FAST_START_TRANSACTIONS where INST_ID = USERENV('Instance')

V$FILEMETRIC
SELECT begin_time, end_time, intsize_csec, file_id, creation_time, average_read_time, average_write_time, physical_reads, physical_writes, physical_block_reads, physical_block_writes FROM gv$filemetric WHERE inst_id = USERENV('INSTANCE')

VIEW_NAME

VIEW_DEFINITION

V$FILEMETRIC_HISTORY
SELECT begin_time, end_time, intsize_csec, file_id, creation_time, average_read_time, average_write_time, physical_reads, physical_writes, physical_block_reads, physical_block_writes FROM gv$filemetric_history WHERE inst_id = USERENV('INSTANCE')

V$FILESPACE_USAGE
SELECT TABLESPACE_ID, RFNO, ALLOCATED_SPACE, FILE_SIZE, FILE_MAXSIZE,

VIEW_NAME

VIEW_DEFINITION

CHANGESCN_BASE, CHANGESCN_WRAP, FLAG FROM GV$FILESPACE_USAGE WHERE INST_ID = USERENV('INSTANCE')

V$FILESTAT
select FILE# , PHYRDS , PHYWRTS , PHYBLKRD , PHYBLKWRT , SINGLEBLKRDS, READTIM , WRITETIM, SINGLEBLKRDTIM, AVGIOTIM, LSTIOTIM, MINIOTIM, MAXIORTM, MAXIOWTM

from GV$FILESTAT where inst_id = USERENV('Instance')

V$FILE_CACHE_TRANSFER

VIEW_NAME

VIEW_DEFINITION
---
select file_number, x_2_null,
x_2_null_forced_write, x_2_null_forced_stale, x_2_s,
x_2_s_forced_write, s_2_null,
s_2_null_forced_stale, rbr, rbr_forced_write,
rbr_forced_stale, null_2_x, s_2_x, null_2_s,
cr_transfers, cur_transfers from gv$file_cache_transfer
where inst_id = USERENV('Instance')

V$FILE_HISTOGRAM

VIEW_NAME

VIEW_DEFINITION
---
select FILE#, SINGLEBLKRDTIM_MILLI, SINGLEBLKRDS from GV$FILE_HISTOGRAM where
inst_id = USERENV('Instance')

V$FILE_PING
select file_number, frequency, x_2_null,
x_2_null_forced_write, x_2_null_forced_stale, x_2_s,
x_2_s_forced_write, x_2_ssx, x_2_ssx_forced_write, s_2_null,
s_2_null_forced_stale, ss_2_null, ss_2_rls, wrb, wrb_forced_write, rbr,
rbr_forced_write, rbr_forced_stale, cbr, cbr_forced_write, null_2_x, s_2_x,

VIEW_NAME

VIEW_DEFINITION
---
ssx_2_x, null_2_s, null_2_ss, op_2_ss from gv$file_ping
where inst_id = USERENV('Instance')

V$FIXED_TABLE
select NAME , OBJECT_ID , TYPE , TABLE_NUM from GV$FIXED_TABLE where inst_id =
USERENV('Instance')

V$FIXED_VIEW_DEFINITION
select VIEW_NAME , VIEW_DEFINITION from GV$FIXED_VIEW_DEFINITION where inst_id

VIEW_NAME

VIEW_DEFINITION

= USERENV('Instance')

V$FLASHBACK_DATABASE_LOG
select OLDEST_FLASHBACK_SCN, OLDEST_FLASHBACK_TIME, RETENTION_TARGET, FLASHBACK_SIZE, ESTIMATED_FLASHBACK_SIZE from GV$FLASHBACK_DATABASE_LOG where inst_id = USERENV('Instance')

V$FLASHBACK_DATABASE_LOGFILE
select NAME, LOG#, THREAD#, SEQUENCE#, BYTES, FIRST_CHANGE#,

VIEW_NAME

VIEW_DEFINITION

FIRST_TIME from GV$FLASHBACK_DATABASE_LOGFILE where inst_id = USERENV('Instance')

V$FLASHBACK_DATABASE_STAT
select BEGIN_TIME, END_TIME, FLASHBACK_DATA, DB_DATA, REDO_DATA, ESTIMATED_FLASHBACK_SIZE from GV$FLASHBACK_DATABASE_STAT where inst_id = USERENV('Instance')

V$FLASHBACK_TXN_GRAPH

VIEW_NAME

VIEW_DEFINITION
---
select COMPENSATING_XID, COMPENSATING_TXN_NAME, XID, TXN_NAME, PARENT_XID, INTERESTING, ORIGINAL, BACKOUT_SEQ, NUM_PREDS, NUM_SUCCS, DEP_XID, DEP_TXN_NAME, TXN_CONF_SQL_ID, DEP_TXN_CONF_SQL_ID from X$KTFTBTXNGRAPH

V$FLASHBACK_TXN_MODS
select COMPENSATING_XID, COMPENSATING_TXN_NAME, XID, TXN_NAME, PARENT_XID, INTERESTING, ORIGINAL, BACKOUT_SEQ,

VIEW_NAME

VIEW_DEFINITION

UNDO_SQL, UNDO_SQL_SQN, UNDO_SQL_SUB_SQN, BACKOUT_SQL_ID, OPERATION, BACKEDOUT, CONFLICT_MOD, MODS_PER_LCR from X$KTFTBTXNMODS

V$FLASH_RECOVERY_AREA_USAGE
select fusg.file_type,
decode(nvl2(ra.name, ra.space_limit, 0), 0, 0,
round(nvl(fusg.space_used, 0)/ra.space_limit, 4) *
100),

VIEW_NAME

```
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(nvl2(ra.name, ra.space_limit, 0), 0, 0,
round(nvl(fusg.space_reclaimable, 0)/ra.space_limit, 4)               *
100),                                          nvl2(ra.name,
fusg.number_of_files, 0)                     from v$recovery_file_dest
ra,                                    (select 'CONTROL FILE'
file_type,                          sum(case when ceilasm = 1 and name
like '+%'                           then
ceil(((block_size*file_size_blks)+1)/1048576)*1048576              else
block_size*file_size_blks                                    end)

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
space_used,                           0
space_reclaimable,                      count(*)
number_of_files                     from v$controlfile,
(select /*+ no_merge */ ceilasm from x$krasga)                  where
is_recovery_dest_file = 'YES'                                union all
select 'REDO LOG'                   file_type,
sum(case when ceilasm = 1 and member like '+%'
then
ceil((l.bytes+1)/1048576)*1048576                                 else

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
l.bytes                                                    end)
space_used,                           0
space_reclaimable,                      count(*)
number_of_files                     from (select group#, bytes from v$log
union                                                       select
group#, bytes from v$standby_log) l, v$logfile lf,              (select /*+
no_merge */ ceilasm from x$krasga)                   where l.group# =
lf.group#                                        and
lf.is_recovery_dest_file = 'YES'                            union all

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select 'ARCHIVED LOG'                 file_type,
sum(al.file_size)              space_used,
sum(case when dl.rectype = 11 then al.file_size
else 0 end)           space_reclaimable,                    count(*)
number_of_files                     from (select recid,
case when ceilasm = 1 and name like '+%'                          then
ceil(((blocks*block_size)+1)/1048576)*1048576                      else
blocks * block_size                                         end
```

file_size from

VIEW_NAME

VIEW_DEFINITION
---
v$archived_log,
(select /*+ no_merge */ ceilasm from x$krasga) where
is_recovery_dest_file = 'YES' and name
is not null) al,

V$FOREIGN_ARCHIVED_LOG
select RECID , STAMP , NAME , DEST_ID , THREAD# , SEQUENCE# ,
RESETLOGS_CHANGE# , RESETLOGS_TIME , RESETLOGS_ID , FIRST_CHANGE# , FIRST_TIME
, NEXT_CHANGE# , NEXT_TIME , BLOCKS , BLOCK_SIZE , CREATOR, REGISTRAR ,

VIEW_NAME

VIEW_DEFINITION

ARCHIVED , APPLIED , DELETED , STATUS , COMPLETION_TIME , DICTIONARY_BEGIN ,
DICTIONARY_END , END_OF_REDO, ARCHIVAL_THREAD#, IS_RECOVERY_DEST_FILE,
COMPRESSED, FAL, END_OF_REDO_TYPE, SOURCE_DBID from GV$FOREIGN_ARCHIVED_LOG
where inst_id = USERENV('Instance')

V$FS_FAILOVER_HISTOGRAM
select REDO_LATENCY,FREQUENCY,LAST_TIME from GV$FS_FAILOVER_HISTOGRAM where
inst_id=USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$FS_FAILOVER_STATS
select LAST_FAILOVER_TIME, LAST_FAILOVER_REASON from GV$FS_FAILOVER_STATS where
inst_id=USERENV('Instance')

V$GCSHVMASTER_INFO
select HV_ID, CURRENT_MASTER, PREVIOUS_MASTER, REMASTER_CNT from
GV$GCSHVMASTER_INFO where inst_id = USERENV('Instance')

V$GCSPFMASTER_INFO

VIEW_NAME

VIEW_DEFINITION
---
select FILE_ID, OBJECT_ID, TYPE, CURRENT_MASTER, PREVIOUS_MASTER, REMASTER_CNT
from GV$GCSPFMASTER_INFO where inst_id = USERENV('Instance')

V$GC_ELEMENT

select gc_element_addr, indx, class, gc_element_name, mode_held,
block_count, releasing, acquiring, writing, recovering, local,
flags from gv$gc_element where inst_id =
USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$GC_ELEMENTS_WITH_COLLISIONS
select gc_element_addr from gv$gc_elements_with_collisions where
inst_id = USERENV('Instance')

V$GES_BLOCKING_ENQUEUE
select HANDLE, GRANT_LEVEL, REQUEST_LEVEL, RESOURCE_NAME1,
RESOURCE_NAME2, PID, TRANSACTION_ID0, TRANSACTION_ID1, GROUP_ID,
OPEN_OPT_DEADLOCK, OPEN_OPT_PERSISTENT, OPEN_OPT_PROCESS_OWNED,
OPEN_OPT_NO_XID, CONVERT_OPT_GETVALUE, CONVERT_OPT_PUTVALUE,

VIEW_NAME

VIEW_DEFINITION

CONVERT_OPT_NOVALUE, CONVERT_OPT_DUBVALUE, CONVERT_OPT_NOQUEUE,
CONVERT_OPT_EXPRESS, CONVERT_OPT_NODEADLOCKWAIT,
CONVERT_OPT_NODEADLOCKBLOCK, WHICH_QUEUE, STATE, AST_EVENT0,
OWNER_NODE, BLOCKED, BLOCKER from GV$GES_BLOCKING_ENQUEUE
where INST_ID = USERENV('Instance')

V$GES_ENQUEUE
select HANDLE, GRANT_LEVEL, REQUEST_LEVEL, RESOURCE_NAME1,
RESOURCE_NAME2, PID, TRANSACTION_ID0, TRANSACTION_ID1, GROUP_ID,

VIEW_NAME

VIEW_DEFINITION

OPEN_OPT_DEADLOCK, OPEN_OPT_PERSISTENT, OPEN_OPT_PROCESS_OWNED,
OPEN_OPT_NO_XID, CONVERT_OPT_GETVALUE, CONVERT_OPT_PUTVALUE,
CONVERT_OPT_NOVALUE, CONVERT_OPT_DUBVALUE, CONVERT_OPT_NOQUEUE,
CONVERT_OPT_EXPRESS, CONVERT_OPT_NODEADLOCKWAIT,
CONVERT_OPT_NODEADLOCKBLOCK, WHICH_QUEUE, STATE, AST_EVENT0,
OWNER_NODE, BLOCKED, BLOCKER from GV$GES_ENQUEUE where INST_ID =
USERENV('Instance')

V$GLOBALCONTEXT

VIEW_NAME

VIEW_DEFINITION
---

select namespace, attribute, value, username,clientidentifier
from gv$globalcontext

V$GLOBAL_BLOCKED_LOCKS
select ADDR , KADDR , SID , TYPE , ID1 , ID2 , LMODE , REQUEST , CTIME from
gv$global_blocked_locks where inst_id = userenv('instance')

V$GLOBAL_TRANSACTION
select FORMATID, GLOBALID, BRANCHID, BRANCHES, REFCOUNT, PREPARECOUNT, STATE,

VIEW_NAME

VIEW_DEFINITION

FLAGS, COUPLING from
GV$GLOBAL_TRANSACTION where
INST_ID = USERENV('Instance')

V$HM_CHECK
select ID, NAME, CLSID, CLS_NAME, FLAGS,
INTERNAL_CHECK, OFFLINE_CAPABLE, DESCRIPTION from GV$HM_CHECK where
inst_id=USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$HM_CHECK_PARAM
select ID, NAME, CHECK_ID, TYPE, DEFAULT_VALUE,
FLAGS, DESCRIPTION from GV$HM_CHECK_PARAM where
inst_id=USERENV('Instance')

V$HM_FINDING
select FINDING_ID, RUN_ID, NAME, PARENT_ID, CHILD_COUNT,
CLASS_NAME, TIME_DETECTED, MODIFIED, PRIORITY, STATUS,
TYPE, DESCRIPTION, DAMAGE_DESCRIPTION from GV$HM_FINDING where

VIEW_NAME

VIEW_DEFINITION
---
inst_id=USERENV('Instance')

V$HM_INFO
select ID, TYPE, NAME, VALUE from GV$HM_INFO where
inst_id=USERENV('Instance')

V$HM_RECOMMENDATION
select RECOMMENDATION_ID, FDG_ID, RUN_ID, NAME, TYPE,
RANK, TIME_DETECTED, EXECUTED, STATUS, DESCRIPTION,

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
REPAIR_SCRIPT from GV$HM_RECOMMENDATION where inst_id=USERENV('Instance')

V$HM_RUN
select RUN_ID,       NAME,       CHECK_NAME,       RUN_MODE,       TIMEOUT,
START_TIME,       LAST_RESUME_TIME,       END_TIME,       MODIFIED_TIME,
STATUS,       SRC_INCIDENT,       NUM_INCIDENT,       ERROR_NUMBER,
PROBLEM_ID  from GV$HM_RUN where inst_id=USERENV('Instance')

V$HS_AGENT

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select unique AGENT_ID, MACHINE, PROCESS, PROGRAM,         OSUSER, STARTTIME,
AGENT_TYPE, FDS_CLASS_ID, FDS_INST_ID         from GV$HS_AGENT where INST_ID =
USERENV('Instance')

V$HS_PARAMETER
select HS_SESSION_ID, PARAMETER, VALUE,  SOURCE, ENV         from
GV$HS_PARAMETER WHERE INST_ID = userenv('instance')

V$HS_SESSION

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select        HS_SESSION_ID, AGENT_ID, SID, DB_LINK, DB_LINK_OWNER, STARTTIME
from GV$HS_SESSION where INST_ID = USERENV('Instance')

V$HVMASTER_INFO
select HV_ID, CURRENT_MASTER, PREVIOUS_MASTER, REMASTER_CNT from
GV$HVMASTER_INFO where inst_id = USERENV('Instance')

V$INCMETER_CONFIG
select TRANSIENT_INCIDENT_LIFETIME,       CRITICAL_FACTOR,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
WARNING_FACTOR,       WEIGHT_FACTOR,       MODIFICATION_TIME from
GV$INCMETER_CONFIG where inst_id=USERENV('Instance')

V$INCMETER_INFO
select INCIDENT_ID,       CREATE_TIME,       IS_DISABLED,       IS_ACTIVE,
IMPT_NATURE,       IMPACT1,       IMPACT2,       IMPACT3,       IMPACT4 from
```

GV$INCMETER_INFO where inst_id=USERENV('Instance')

V$INCMETER_SUMMARY

VIEW_NAME

VIEW_DEFINITION
---
select SEVERITY_INDEX, CRITICAL_INICDENTS, WARNING_INCIDENTS, LAST_HOUR_INCIDENTS, CREATE_TIME, OLDEST_TRANSIENT_INC_CTIME, OLDEST_PERSISTENT_INC_CTIME, LATEST_INC_CTIME from GV$INCMETER_SUMMARY where inst_id=USERENV('Instance')

V$INDEXED_FIXED_COLUMN
select TABLE_NAME , INDEX_NUMBER , COLUMN_NAME , COLUMN_POSITION from GV$INDEXED_FIXED_COLUMN where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$INSTANCE
select INSTANCE_NUMBER , INSTANCE_NAME , HOST_NAME , VERSION , STARTUP_TIME , STATUS , PARALLEL , THREAD# , ARCHIVER , LOG_SWITCH_WAIT , LOGINS , SHUTDOWN_PENDING, DATABASE_STATUS, INSTANCE_ROLE, ACTIVE_STATE, BLOCKED from GV$INSTANCE where inst_id = USERENV('Instance')

V$INSTANCE_CACHE_TRANSFER
select instance, class, lost, lost_time, cr_block, cr_block_time, cr_2hop, cr_2hop_time, cr_3hop, cr_3hop_time, cr_busy, cr_busy_time, cr_congested,

VIEW_NAME

VIEW_DEFINITION
---
cr_congested_time, current_block, current_block_time, current_2hop, current_2hop_time, current_3hop, current_3hop_time, current_busy, current_busy_time, current_congested, current_congested_time from gv$instance_cache_transfer where inst_id = USERENV('Instance')

V$INSTANCE_LOG_GROUP
select THREAD# , STATUS , ENABLED , GROUPS , INSTANCE , OPEN_TIME , CURRENT_GROUP# , SEQUENCE# , CHECKPOINT_CHANGE# , CHECKPOINT_TIME , ENABLE_CHANGE# , ENABLE_TIME , DISABLE_CHANGE# , DISABLE_TIME from

VIEW_NAME

VIEW_DEFINITION

GV$INSTANCE_LOG_GROUP where inst_id = USERENV('Instance')

V$INSTANCE_RECOVERY
select RECOVERY_ESTIMATED_IOS, ACTUAL_REDO_BLKS, TARGET_REDO_BLKS,
LOG_FILE_SIZE_REDO_BLKS, LOG_CHKPT_TIMEOUT_REDO_BLKS,
LOG_CHKPT_INTERVAL_REDO_BLKS, FAST_START_IO_TARGET_REDO_BLKS,
TARGET_MTTR, ESTIMATED_MTTR, CKPT_BLOCK_WRITES, OPTIMAL_LOGFILE_SIZE,
ESTD_CLUSTER_AVAILABLE_TIME, WRITES_MTTR, WRITES_LOGFILE_SIZE,
WRITES_LOG_CHECKPOINT_SETTINGS, WRITES_OTHER_SETTINGS, WRITES_AUTOTUNE,

VIEW_NAME

VIEW_DEFINITION

WRITES_FULL_THREAD_CKPT from GV$INSTANCE_RECOVERY where INST_ID =
USERENV('Instance')

V$IOFUNCMETRIC
SELECT begin_time, end_time, intsize_csec, function_id,
function_name, small_read_mbps, small_write_mbps,
large_read_mbps, large_write_mbps, small_read_iops,
small_write_iops, large_read_iops, large_write_iops,
avg_wait_time FROM gv$iofuncmetric WHERE inst_id =

VIEW_NAME

VIEW_DEFINITION

USERENV('INSTANCE')

V$IOFUNCMETRIC_HISTORY
SELECT begin_time, end_time, intsize_csec, function_id,
function_name, small_read_mbps, small_write_mbps,
large_read_mbps, large_write_mbps, small_read_iops,
small_write_iops, large_read_iops, large_write_iops,
avg_wait_time FROM gv$iofuncmetric_history WHERE inst_id =
USERENV('INSTANCE')

VIEW_NAME

VIEW_DEFINITION

V$IOSTAT_CONSUMER_GROUP
SELECT CONSUMER_GROUP_ID, SMALL_READ_MEGABYTES,
SMALL_WRITE_MEGABYTES, LARGE_READ_MEGABYTES,
LARGE_WRITE_MEGABYTES, SMALL_READ_REQS,
SMALL_WRITE_REQS, LARGE_READ_REQS,
LARGE_WRITE_REQS, NUMBER_OF_WAITS, WAIT_TIME
FROM GV$IOSTAT_CONSUMER_GROUP where inst_id=USERENV('Instance')

VIEW_NAME

```
VIEW_DEFINITION
--------------------------------------------------------------------------------
V$IOSTAT_FILE
SELECT FILE_NO, FILETYPE_ID, FILETYPE_NAME,
SMALL_READ_MEGABYTES, SMALL_WRITE_MEGABYTES,
LARGE_READ_MEGABYTES, LARGE_WRITE_MEGABYTES,
SMALL_READ_REQS, SMALL_WRITE_REQS,
SMALL_SYNC_READ_REQS, LARGE_READ_REQS,
LARGE_WRITE_REQS, SMALL_READ_SERVICETIME,
SMALL_WRITE_SERVICETIME, SMALL_SYNC_READ_LATENCY,
LARGE_READ_SERVICETIME, LARGE_WRITE_SERVICETIME,                ASYNCH_IO,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
RETRIES_ON_ERROR                                FROM GV$IOSTAT_FILE
where inst_id=USERENV('Instance')

V$IOSTAT_FUNCTION
SELECT FUNCTION_ID, FUNCTION_NAME,
SMALL_READ_MEGABYTES, SMALL_WRITE_MEGABYTES,
LARGE_READ_MEGABYTES, LARGE_WRITE_MEGABYTES, SMALL_READ_REQS,
SMALL_WRITE_REQS, LARGE_READ_REQS, LARGE_WRITE_REQS,
NUMBER_OF_WAITS, WAIT_TIME                                      FROM

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$IOSTAT_FUNCTION where inst_id=USERENV('Instance')

V$IOSTAT_NETWORK
select  CLIENT,       READS#,       WRITES#,       KBYTES_READ,
KBYTES_WRITTEN,       READ_LATENCY,       WRITE_LATENCY from GV$IOSTAT_NETWORK
where inst_id=USERENV('Instance')

V$IO_CALIBRATION_STATUS
SELECT status, calibration_time                         FROM

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$IO_CALIBRATION_STATUS                         WHERE  inst_id =
USERENV('INSTANCE')

V$IR_FAILURE
select FAILURE_ID,       PARENT_ID,       CHILD_COUNT,       CLASS_NAME,
TIME_DETECTED,       MODIFIED,       DESCRIPTION,       IMPACTS,       PRIORITY,
STATUS from GV$IR_FAILURE where inst_id=USERENV('Instance')
```

V$IR_FAILURE_SET

VIEW_NAME

VIEW_DEFINITION
----
select ADVISE_ID, FAILURE_ID, MANUAL_REPAIRS_ONLY from
GV$IR_FAILURE_SET where inst_id=USERENV('Instance')

V$IR_MANUAL_CHECKLIST
select ADVISE_ID, RANK, REQUIRED, MESSAGE from
GV$IR_MANUAL_CHECKLIST where inst_id=USERENV('Instance')

V$IR_REPAIR
select REPAIR_ID, ADVISE_ID, SUMMARY, RANK,

VIEW_NAME

VIEW_DEFINITION

TIME_DETECTED, EXECUTED, ESTIMATED_DATA_LOSS,
DETAILED_DESCRIPTION, REPAIR_SCRIPT, ESTIMATED_REPAIR_TIME,
ACTUAL_REPAIR_TIME, STATUS from GV$IR_REPAIR where
inst_id=USERENV('Instance')

V$JAVAPOOL
select CATEGORY, MEMUSED from gv$javapool where inst_id = USERENV('Instance')

V$JAVA_LIBRARY_CACHE_MEMORY

VIEW_NAME

VIEW_DEFINITION
----
select lc_namespace, lc_inuse_memory_objects, lc_inuse_memory_size,
lc_freeable_memory_objects, lc_freeable_memory_size from
gv$java_library_cache_memory where inst_id = USERENV('Instance')

V$JAVA_POOL_ADVICE
select java_pool_size_for_estimate, java_pool_size_factor, estd_lc_size,
estd_lc_memory_objects, estd_lc_time_saved, estd_lc_time_saved_factor,
estd_lc_load_time, estd_lc_load_time_factor, estd_lc_memory_object_hits from
gv$java_pool_advice where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$LATCH
select addr,latch#,level#,name,hash,gets,misses,sleeps,immediate_gets,
immediate_misses,waiters_woken,waits_holding_latch,spin_gets,

sleep1,sleep2,sleep3,sleep4,sleep5,sleep6,sleep7,sleep8,sleep9,
sleep10,sleep11,wait_time from gv$latch where inst_id = USERENV('Instance')

V$LATCHHOLDER
select PID , SID , LADDR , NAME , GETS from GV$LATCHHOLDER where inst_id =

VIEW_NAME

VIEW_DEFINITION

USERENV('Instance')

V$LATCHNAME
select latch#,name, hash from gv$latchname where inst_id = userenv('Instance')

V$LATCH_CHILDREN
select ADDR , LATCH# , CHILD# , LEVEL# , NAME , HASH , GETS , MISSES , SLEEPS ,
IMMEDIATE_GETS , IMMEDIATE_MISSES , WAITERS_WOKEN , WAITS_HOLDING_LATCH ,
SPIN_GETS , SLEEP1 , SLEEP2 , SLEEP3 , SLEEP4 , SLEEP5 , SLEEP6 , SLEEP7 ,

VIEW_NAME

VIEW_DEFINITION

SLEEP8 , SLEEP9 , SLEEP10 , SLEEP11 , WAIT_TIME from GV$LATCH_CHILDREN where
inst_id = USERENV('Instance')

V$LATCH_MISSES
select PARENT_NAME, LOCATION, NWFAIL_COUNT, SLEEP_COUNT, WTR_SLP_COUNT,
LONGHOLD_COUNT, LOCATION from GV$LATCH_MISSES where inst_id =
USERENV('Instance')

V$LATCH_PARENT

VIEW_NAME

VIEW_DEFINITION
----
select ADDR , LATCH# , LEVEL# , NAME , HASH , GETS , MISSES , SLEEPS ,
IMMEDIATE_GETS , IMMEDIATE_MISSES , WAITERS_WOKEN , WAITS_HOLDING_LATCH ,
SPIN_GETS , SLEEP1 , SLEEP2 , SLEEP3 , SLEEP4 , SLEEP5 , SLEEP6 , SLEEP7 ,
SLEEP8 , SLEEP9 , SLEEP10 , SLEEP11 , WAIT_TIME from GV$LATCH_PARENT where
inst_id = USERENV('Instance')

V$LIBRARYCACHE
select NAMESPACE , GETS , GETHITS , GETHITRATIO , PINS , PINHITS , PINHITRATIO
, RELOADS , INVALIDATIONS , DLM_LOCK_REQUESTS , DLM_PIN_REQUESTS ,

VIEW_NAME

VIEW_DEFINITION

DLM_PIN_RELEASES , DLM_INVALIDATION_REQUESTS , DLM_INVALIDATIONS from
GV$LIBRARYCACHE where inst_id = USERENV('Instance')

V$LIBRARY_CACHE_MEMORY
select lc_namespace, lc_inuse_memory_objects, lc_inuse_memory_size,
lc_freeable_memory_objects, lc_freeable_memory_size from
gv$library_cache_memory where inst_id = USERENV('Instance')

V$LICENSE

VIEW_NAME

VIEW_DEFINITION
----
select sessions_max,sessions_warning,sessions_current,sessions_highwater,
users_max, cpu_count_current, cpu_core_count_current, cpu_socket_count_current,
cpu_count_highwater, cpu_core_count_highwater, cpu_socket_count_highwater from
gv$license where inst_id = userenv('Instance')

V$LOADISTAT
select OWNER, TABNAME, INDEXNAME, SUBNAME, MESSAGE_NUM, MESSAGE from
GV$LOADISTAT where INST_ID = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$LOADPSTAT
select OWNER , TABNAME , PARTNAME , LOADED from GV$LOADPSTAT where inst_id =
USERENV('Instance')

V$LOBSTAT
select inst_id, LOBTSN, LOBRDBA, LOBOBJID, LOBCURRTIME, LOBEXPMQL, LOBSQLMQL,
LOBSPCANALTIME,   LOBUNDORETTIME from gv$lobstat where inst_id =
userenv('instance')

VIEW_NAME

VIEW_DEFINITION

V$LOCK
select ADDR , KADDR , SID , TYPE , ID1 , ID2 , LMODE , REQUEST , CTIME , BLOCK
from GV$LOCK where inst_id = USERENV('Instance')

V$LOCKED_OBJECT
select xidusn,xidslot,xidsqn,object_id,session_id,oracle_username,
os_user_name,process,locked_mode from gv$locked_object where inst_id =
USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$LOCKS_WITH_COLLISIONS
select lock_element_addr from v$bh where (forced_writes +
forced_reads) > 10 group by lock_element_addr
having count(*) >= 2

V$LOCK_ACTIVITY
SELECT FROM_VAL,TO_VAL,ACTION_VAL,COUNTER FROM GV$LOCK_ACTIVITY where
INST_ID = USERENV('INSTANCE')

VIEW_NAME

VIEW_DEFINITION

V$LOCK_ELEMENT
select lock_element_addr, indx, class, lock_element_name, mode_held,
block_count, releasing, acquiring, invalid, flags from gv$lock_element
where inst_id = USERENV('Instance')

V$LOCK_TYPE
select TYPE, NAME, ID1_TAG, ID2_TAG, IS_USER, DESCRIPTION from GV$LOCK_TYPE
where INST_ID = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$LOG
select GROUP# , THREAD# , SEQUENCE# , BYTES , MEMBERS , ARCHIVED , STATUS ,
FIRST_CHANGE# , FIRST_TIME from GV$LOG where inst_id = USERENV('Instance')

V$LOGFILE
select GROUP# , STATUS , TYPE , MEMBER, IS_RECOVERY_DEST_FILE from GV$LOGFILE
where inst_id = USERENV('Instance')

V$LOGHIST

VIEW_NAME

VIEW_DEFINITION
---
select THREAD# , SEQUENCE# , FIRST_CHANGE# , FIRST_TIME , SWITCH_CHANGE# from
GV$LOGHIST where inst_id = USERENV('Instance')

V$LOGMNR_CALLBACK
select session_id, function_id, description, type from
gv$logmnr_callback where inst_id = userenv('instance')

V$LOGMNR_CONTENTS
select SCN, START_SCN, COMMIT_SCN, TIMESTAMP, START_TIMESTAMP,

VIEW_NAME

VIEW_DEFINITION

COMMIT_TIMESTAMP, XIDUSN, XIDSLT, XIDSQN, XID, PXIDUSN, PXIDSLT,
PXIDSQN, PXID, TX_NAME, OPERATION, OPERATION_CODE, ROLLBACK,
SEG_OWNER, SEG_NAME, TABLE_NAME, SEG_TYPE, SEG_TYPE_NAME, TABLE_SPACE,
ROW_ID, USERNAME, OS_USERNAME, MACHINE_NAME, AUDIT_SESSIONID,
SESSION#, SERIAL#, SESSION_INFO, THREAD#, SEQUENCE#, RBASQN, RBABLK,
RBABYTE, UBAFIL, UBABLK, UBAREC, UBASQN, ABS_FILE#,REL_FILE#,
DATA_BLK#, DATA_OBJ#, DATA_OBJV#, DATA_OBJD#, SQL_REDO,
SQL_UNDO, RS_ID, SSN, CSF, INFO, STATUS, REDO_VALUE, UNDO_VALUE,
SAFE_RESUME_SCN, CSCN, OBJECT_ID from GV$LOGMNR_CONTENTS where inst_id =

VIEW_NAME

VIEW_DEFINITION
---
userenv('instance')

V$LOGMNR_DBA_OBJECTS
select OWNER, OBJECT_NAME, SUBOBJECT_NAME, OBJECT_ID, DATA_OBJECT_ID,
OBJECT_TYPE, CREATED, LAST_DDL_TIME, TIMESTAMP, STATUS, TEMPORARY,
GENERATED, SECONDARY from gv$logmnr_dba_objects where inst_id =
userenv('instance')

V$LOGMNR_DICTIONARY

VIEW_NAME

VIEW_DEFINITION
---
select DB_NAME, DB_ID, DB_CREATED, TIMESTAMP, RESET_SCN,
RESET_SCN_TIME, DB_VERSION_TIME, DB_CHARACTER_SET, DB_VERSION,
DB_STATUS, DICTIONARY_SCN, ENABLED_THREAD_MAP, DB_TXN_SCN,
FILENAME, INFO, STATUS from GV$LOGMNR_DICTIONARY where inst_id =
userenv('instance')

V$LOGMNR_DICTIONARY_LOAD
select session_id, logmnr_uid, action#,
opcode, command, current_state, completed_actions,

VIEW_NAME

VIEW_DEFINITION
---
total_actions, loaded, percent_done from gv$logmnr_dictionary_load
where inst_id = userenv('instance')

V$LOGMNR_EXTENTS
select OWNER, SEGMENT_NAME, PARTITION_NAME, SEGMENT_TYPE, TABLESPACE_NAME,
EXTENT_ID, FILE_ID, BLOCK_ID, BYTES, BLOCKS, RELATIVE_FNO from GV$LOGMNR_EXTENTS
where inst_id = userenv('instance')

V$LOGMNR_LATCH

VIEW_NAME

VIEW_DEFINITION
----
select session_id, name, child_addr, state from
gv$logmnr_latch where inst_id = userenv('instance')

V$LOGMNR_LOGFILE
select log_id, filename, low_time, next_time, db_id, db_name,
reset_scnwrp, reset_scnbas, reset_scn_time, thread_id,
thread_sqn, low_scnwrp, low_scnbas, next_scnwrp, next_scnbas,
file_state from gv$logmnr_logfile where inst_id = userenv('instance')

VIEW_NAME

VIEW_DEFINITION

V$LOGMNR_LOGS
select LOG_ID, FILENAME, LOW_TIME, HIGH_TIME, DB_ID, DB_NAME,
RESET_SCN, RESET_SCN_TIME, THREAD_ID, THREAD_SQN, LOW_SCN,
NEXT_SCN, DICTIONARY_BEGIN, DICTIONARY_END,
TYPE, BLOCKSIZE, FILESIZE, INFO, STATUS from GV$LOGMNR_LOGS where inst_id
= userenv('instance')

V$LOGMNR_OBJECT_SEGMENTS
select OWNER, SEGMENT_NAME, PARTITION_NAME, SEGMENT_TYPE,

VIEW_NAME

VIEW_DEFINITION

SEGMENT_TYPE_ID, TABLESPACE_ID, TABLESPACE_NAME, BLOCKSIZE, HEADER_FILE,
HEADER_BLOCK, BYTES, BLOCKS, EXTENTS, INITIAL_EXTENT, NEXT_EXTENT,
MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE, FREELISTS, FREELIST_GROUPS,
RELATIVE_FNO, BUFFER_POOL_ID, SEGMENT_FLAGS, SEGMENT_OBJD from
GV$LOGMNR_OBJECT_SEGMENTS where inst_id = userenv('instance')

V$LOGMNR_PARAMETERS
select START_DATE, REQUIRED_START_DATE, END_DATE, START_SCN,
REQUIRED_START_SCN, END_SCN, OPTIONS, INFO, STATUS from

VIEW_NAME

VIEW_DEFINITION

GV$LOGMNR_PARAMETERS where inst_id = userenv('instance')

V$LOGMNR_PROCESS
select session_id, pid, spid, role, username, sid, serial#,
latchwait, latchspin, work_microsec, overhead_microsec from
gv$logmnr_process where inst_id = userenv('instance')

V$LOGMNR_REGION
select memstate, state, owning_process from gv$logmnr_region where

VIEW_NAME

VIEW_DEFINITION
----
inst_id = userenv('instance')

V$LOGMNR_SESSION
select session_id, session_name, session_state, db_name, db_id,
reset_scn, reset_timestamp, num_process, chunk_size,
start_scn, end_scn, spill_scn, processed_scn, prepared_scn,
read_scn, low_mark_scn, consumed_scn,
max_memory_size, used_memory_size, builder_work_size,
prepared_work_size, available_work_size, available_txn,

VIEW_NAME

VIEW_DEFINITION
----
available_committed_txn, delivered_txn, delivered_committed_txn,
pinned_txn, pinned_committed_txn, checkpoint_interval from
GV$LOGMNR_SESSION where inst_id = userenv('instance')

V$LOGMNR_STATS
select session_id, name,value from gv$logmnr_stats where inst_id =
USERENV('Instance')

V$LOGMNR_SYS_DBA_SEGS

VIEW_NAME

VIEW_DEFINITION
----
select OWNER, SEGMENT_NAME, PARTITION_NAME, SEGMENT_TYPE,
SEGMENT_TYPE_ID, TABLESPACE_ID, TABLESPACE_NAME, BLOCKSIZE, HEADER_FILE,
HEADER_BLOCK, BYTES, BLOCKS, EXTENTS, INITIAL_EXTENT, NEXT_EXTENT,
MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE, FREELISTS, FREELIST_GROUPS,
RELATIVE_FNO, BUFFER_POOL_ID, SEGMENT_FLAGS, SEGMENT_OBJD from
gv$logmnr_sys_dba_segs where inst_id = userenv('instance')

V$LOGMNR_SYS_OBJECTS

select OBJECT_TYPE, OBJECT_TYPE_ID, SEGMENT_TYPE_ID, OBJECT_ID, HEADER_FILE,

VIEW_NAME

VIEW_DEFINITION

HEADER_BLOCK, TS_NUMBER from gv$logmnr_sys_objects where inst_id =
userenv('instance')

V$LOGMNR_TRANSACTION
select session_id, xid, xidusn, xidslt, xidsqn, parent_xid,
parent_xidusn, parent_xidslt, parent_xidsqn, start_time,
start_scn, chunk#, total_chunks, redo_thread, low_time, low_scn,
high_time, high_scn, lcr_count, spilled_lcr_count, dflag, mflag,
mflag2, state, type, mining_status, queue from

VIEW_NAME

VIEW_DEFINITION
---
gv$logmnr_transaction where inst_id = userenv('instance')

V$LOGSTDBY
select serial#,logstdby_id,pid,type,status_code,status,high_scn from gv$logstdby
where inst_id = USERENV('Instance')

V$LOGSTDBY_PROCESS
select sid, serial#,logstdby_id,spid,type,status_code,status,high_scn from
gv$logstdby_process where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$LOGSTDBY_PROGRESS
select applied_scn,applied_time,restart_scn,restart_time,latest_scn,
latest_time,mining_scn, mining_time from gv$logstdby_progress where inst_id =
USERENV('Instance')

V$LOGSTDBY_STATE
select primary_dbid,session_id,realtime_apply,state from gv$logstdby_state where
inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$LOGSTDBY_STATS
select name,value from gv$logstdby_stats where inst_id = USERENV('Instance')

V$LOGSTDBY_TRANSACTION
select primary_xidusn, primary_xidslt, primary_xidsqn, primary_xid,
primary_start_scn, primary_start_time, primary_parent_xidusn,
primary_parent_xidslt, primary_parent_xidsqn,
primary_parent_xid, type, mining_status, apply_status, sid,

VIEW_NAME

VIEW_DEFINITION
---
serial# from gv$logstdby_transaction where inst_id =
USERENV('Instance')

V$LOG_HISTORY
select RECID , STAMP , THREAD# , SEQUENCE# , FIRST_CHANGE# , FIRST_TIME ,
NEXT_CHANGE#, RESETLOGS_CHANGE#, RESETLOGS_TIME from GV$LOG_HISTORY

V$MANAGED_STANDBY
select PROCESS, PID, STATUS, CLIENT_PROCESS, CLIENT_PID, CLIENT_DBID, GROUP#,

VIEW_NAME

VIEW_DEFINITION

RESETLOG_ID, THREAD#, SEQUENCE#, BLOCK#, BLOCKS, DELAY_MINS, KNOWN_AGENTS,
ACTIVE_AGENTS from GV$MANAGED_STANDBY where inst_id = USERENV('Instance')

V$MAP_COMP_LIST
select ELEM_IDX,NUM_COMP,COMP1_NAME,COMP1_VAL,
COMP2_NAME,COMP2_VAL,COMP3_NAME,COMP3_VAL,
COMP4_NAME,COMP4_VAL,COMP5_NAME,COMP5_VAL from gv$map_comp_list where inst_id =
USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$MAP_ELEMENT
select ELEM_NAME,ELEM_IDX,ELEM_CFGID,ELEM_TYPE,
ELEM_SIZE,ELEM_NSUBELEM,ELEM_DESCR,STRIPE_SIZE,LIB_IDX from gv$map_element where
inst_id = USERENV('Instance')

V$MAP_EXT_ELEMENT
select ELEM_IDX,NUM_ATTRB,ATTRB1_NAME,ATTRB1_VAL,
ATTRB2_NAME,ATTRB2_VAL,ATTRB3_NAME,ATTRB3_VAL,
ATTRB4_NAME,ATTRB4_VAL,ATTRB5_NAME,ATTRB5_VAL from gv$map_ext_element where

VIEW_NAME

VIEW_DEFINITION
---

inst_id = USERENV('Instance')

V$MAP_FILE
select FILE_MAP_IDX,FILE_CFGID,FILE_STATUS,FILE_NAME,
FILE_TYPE,FILE_STRUCTURE,FILE_SIZE,FILE_NEXTS,LIB_IDX from gv$map_file where
inst_id = USERENV('Instance')

V$MAP_FILE_EXTENT
select FILE_MAP_IDX,EXT_NUM,EXT_ELEM_OFF,EXT_SIZE,

VIEW_NAME

VIEW_DEFINITION

EXT_FILE_OFF,EXT_TYPE,ELEM_IDX from gv$map_file_extent where inst_id =
USERENV('Instance')

V$MAP_FILE_IO_STACK
select FILE_MAP_IDX,DEPTH,ELEM_IDX,CU_SIZE,
STRIDE,NUM_CU,ELEM_OFFSET,FILE_OFFSET,DATA_TYPE,
PARITY_POS,PARITY_PERIOD,ID,PARENT_ID from gv$map_file_io_stack where inst_id =
USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$MAP_LIBRARY
select LIB_IDX,LIB_NAME,VENDOR_NAME,PROTOCOL_NUM,
VERSION_NUM,PATH_NAME,MAP_FILE,FILE_CFGID,MAP_ELEM, ELEM_CFGID,MAP_SYNC from
gv$map_library where inst_id = USERENV('Instance')

V$MAP_SUBELEMENT
select CHILD_IDX,PARENT_IDX,SUB_NUM,SUB_SIZE, ELEM_OFFSET,SUB_FLAGS from
gv$map_subelement where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$MAX_ACTIVE_SESS_TARGET_MTH
select name from gv$max_active_sess_target_mth where inst_id =
userenv('instance')

V$MEMORY_CURRENT_RESIZE_OPS
select component, oper_type, oper_mode, parameter, initial_size,
target_size, current_size, start_time, last_update_time from
gv$memory_current_resize_ops where inst_id = USERENV('Instance')

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
V$MEMORY_DYNAMIC_COMPONENTS
select component, current_size, min_size, max_size, user_specified_size,
oper_count, last_oper_type, last_oper_mode, last_oper_time, granule_size   from
gv$memory_dynamic_components   where inst_id = USERENV('Instance')

V$MEMORY_RESIZE_OPS
select component, oper_type, oper_mode, parameter, initial_size,
target_size, final_size, status, start_time, end_time   from
gv$memory_resize_ops   where inst_id = USERENV('Instance')

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

V$MEMORY_TARGET_ADVICE
select memory_size, memory_size_factor, estd_db_time,
estd_db_time_factor, version                                 from
gv$memory_target_advice where inst_id = userenv('instance')

V$METRIC
SELECT begin_time, end_time, intsize_csec,          group_id, entity_id,
entity_sequence, metric_id, metric_name,          value, metric_unit

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
FROM   gv$metric        WHERE  inst_id = USERENV('INSTANCE')

V$METRICGROUP
SELECT group_id, name, interval_size, max_interval        FROM
gv$metricgroup        WHERE  inst_id = USERENV('INSTANCE')

V$METRICNAME
SELECT group_id, group_name, metric_id,          metric_name, metric_unit
FROM   gv$metricname        WHERE  inst_id = USERENV('INSTANCE')

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

V$METRIC_HISTORY
SELECT begin_time, end_time, intsize_csec,          group_id, entity_id,
entity_sequence, metric_id, metric_name,          value, metric_unit
FROM   gv$metric_history        WHERE  inst_id = USERENV('INSTANCE')
```

V$MTTR_TARGET_ADVICE
select mttr_target_for_estimate, advice_status, dirty_limit,
estd_cache_writes, estd_cache_write_factor, estd_total_writes,

VIEW_NAME

VIEW_DEFINITION
---
estd_total_write_factor,
estd_total_ios, estd_total_io_factor from gv$mttr_target_advice where
inst_id = userenv('instance')

V$MUTEX_SLEEP
select MUTEX_TYPE, LOCATION, SLEEPS, WAIT_TIME from GV$MUTEX_SLEEP where INST_ID
= USERENV('INSTANCE')

V$MUTEX_SLEEP_HISTORY

VIEW_NAME

VIEW_DEFINITION
---
select MUTEX_IDENTIFIER, SLEEP_TIMESTAMP, MUTEX_TYPE, GETS, SLEEPS,
REQUESTING_SESSION, BLOCKING_SESSION, LOCATION, MUTEX_VALUE, P1, P1RAW, P2, P3,
P4, P5 from GV$MUTEX_SLEEP_HISTORY where INST_ID = USERENV('INSTANCE')

V$MVREFRESH
SELECT SID, SERIAL#, CURRMVOWNER, CURRMVNAME FROM GV$MVREFRESH

V$MYSTAT
select SID , STATISTIC# , VALUE from GV$MYSTAT where inst_id =

VIEW_NAME

VIEW_DEFINITION

USERENV('Instance')

V$NFS_CLIENTS
SELECT NFSCLIENTID, NFSPRINCIPAL, NFSOPAQUECLIENT, NFSVERIFIER,
NFSLEASEEXPIRY, NFSCLIENTNETID || NFSCLIENTADDR,
decode(bitand(NFSFLAGS,1), 1, 'TRUE', 'FALSE') FROM X$NFSCLIENTS

V$NFS_LOCKS
SELECT NFSOPENSTATEID, NFSOPENSEQNO, NFSLOCKSTATEID, NFSLOCKSEQNO,

VIEW_NAME

VIEW_DEFINITION

NFSLOCKOWNER, NFSLOCKOFFSET, NFSLOCKLENGTH, NFSLOCKTYPE FROM
X$NFSLOCKS

V$NFS_OPEN_FILES
SELECT NFSCLIENTID, NFSOPENOWNER, NFSOPENSTATEID, NFSOPENFILEHANDLE,
NFSOPENSEQID, decode(bitand(NFSOPENFLAGS,1), 0, 'FALSE', 'TRUE'),
decode(bitand(NFSOPENFLAGS,2), 0, 'FALSE', 'TRUE'),
decode(bitand(NFSOPENFLAGS,384), 384, 'SharedReadWrite', 128, 'SharedRead', 256,
'SharedWrite'), decode(bitand(NFSOPENFLAGS,1536), 1536,

VIEW_NAME

VIEW_DEFINITION

'DenyReadWrite', 512, 'DenyRead', 1024, 'DenyWrite'),
decode(bitand(NFSOPENFLAGS,64), 0, 'FALSE', 'TRUE') FROM X$NFSOPENS

V$NLS_PARAMETERS
select PARAMETER , VALUE from GV$NLS_PARAMETERS where inst_id =
USERENV('Instance')

V$NLS_VALID_VALUES
select PARAMETER , VALUE, ISDEPRECATED from GV$NLS_VALID_VALUES where inst_id =

VIEW_NAME

VIEW_DEFINITION

USERENV('Instance')

V$OBJECT_DEPENDENCY
select FROM_ADDRESS , FROM_HASH , TO_OWNER , TO_NAME , TO_ADDRESS , TO_HASH ,
TO_TYPE from GV$OBJECT_DEPENDENCY where inst_id = USERENV('Instance')

V$OBJECT_PRIVILEGE
SELECT OBJECT_TYPE_NAME, OBJECT_TYPE_ID, PRIVILEGE_ID, PRIVILEGE_NAME FROM
X$KZPOPR

VIEW_NAME

VIEW_DEFINITION

V$OBSOLETE_PARAMETER
select NAME , ISSPECIFIED from GV$OBSOLETE_PARAMETER where inst_id =
USERENV('Instance')

V$OFFLINE_RANGE
select RECID , STAMP , FILE# , OFFLINE_CHANGE# , ONLINE_CHANGE# , ONLINE_TIME,
RESETLOGS_CHANGE#, RESETLOGS_TIME from GV$OFFLINE_RANGE where inst_id =
USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$OPEN_CURSOR
select SADDR , SID , USER_NAME , ADDRESS , HASH_VALUE, SQL_ID, SQL_TEXT,
LAST_SQL_ACTIVE_TIME, SQL_EXEC_ID from GV$OPEN_CURSOR where inst_id =
USERENV('Instance')

V$OPTION
select PARAMETER , VALUE from GV$OPTION where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$OSSTAT
select STAT_NAME, VALUE, OSSTAT_ID, COMMENTS, CUMULATIVE from GV$OSSTAT where
INST_ID = USERENV('Instance')

V$PARALLEL_DEGREE_LIMIT_MTH
select name from gv$parallel_degree_limit_mth where inst_id =
userenv('instance')

V$PARAMETER

VIEW_NAME

VIEW_DEFINITION
----
select NUM , NAME , TYPE , VALUE , DISPLAY_VALUE, ISDEFAULT , ISSES_MODIFIABLE
, ISSYS_MODIFIABLE , ISINSTANCE_MODIFIABLE, ISMODIFIED , ISADJUSTED ,
ISDEPRECATED, ISBASIC, DESCRIPTION, UPDATE_COMMENT, HASH from GV$PARAMETER
where inst_id = USERENV('Instance')

V$PARAMETER2
select NUM , NAME , TYPE , VALUE , DISPLAY_VALUE, ISDEFAULT , ISSES_MODIFIABLE
, ISSYS_MODIFIABLE , ISINSTANCE_MODIFIABLE, ISMODIFIED , ISADJUSTED ,
ISDEPRECATED, ISBASIC, DESCRIPTION, ORDINAL, UPDATE_COMMENT from GV$PARAMETER2

VIEW_NAME

VIEW_DEFINITION
----
where inst_id = USERENV('Instance')

V$PARAMETER_VALID_VALUES
select num, name, ordinal, value, isdefault from GV$PARAMETER_VALID_VALUES where
INST_ID = USERENV('Instance')

V$PERSISTENT_PUBLISHERS
select queue_id, queue_schema, queue_name, publisher_name, publisher_address,

protocol, enqueued_msgs, elapsed_enqueue_time, last_enqueue_time from

VIEW_NAME

VIEW_DEFINITION
---
gv$persistent_publishers where inst_id = USERENV('Instance')

V$PERSISTENT_QUEUES
select queue_id, queue_schema, queue_name,first_activity_time, enqueued_msgs,
dequeued_msgs, browsed_msgs, elapsed_enqueue_time, elapsed_dequeue_time,
elapsed_transformation_time, elapsed_rule_evaluation_time,
enqueued_expiry_msgs, enqueued_delay_msgs, msgs_made_expired, msgs_made_ready,
last_enqueue_time, last_dequeue_time, last_tm_expiry_time, last_tm_ready_time
from gv$persistent_queues where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$PERSISTENT_SUBSCRIBERS
select queue_id, queue_schema, queue_name, subscriber_id, subscriber_name,
subscriber_address, protocol, subscriber_type, first_activity_time,
enqueued_msgs, dequeued_msgs, browsed_msgs, expired_msgs,
dequeued_msg_latency, last_enqueue_time, last_dequeue_time from
gv$persistent_subscribers where inst_id = USERENV('Instance')

V$PGASTAT

VIEW_NAME

VIEW_DEFINITION
---
select NAME, VALUE, UNIT from GV$PGASTAT where
INST_ID = USERENV('Instance')

V$PGA_TARGET_ADVICE
select PGA_TARGET_FOR_ESTIMATE, PGA_TARGET_FACTOR,
ADVICE_STATUS, BYTES_PROCESSED, ESTD_TIME,
ESTD_EXTRA_BYTES_RW, ESTD_PGA_CACHE_HIT_PERCENTAGE,
ESTD_OVERALLOC_COUNT from GV$PGA_TARGET_ADVICE where INST_ID =
USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$PGA_TARGET_ADVICE_HISTOGRAM
select PGA_TARGET_FOR_ESTIMATE, PGA_TARGET_FACTOR,
ADVICE_STATUS, LOW_OPTIMAL_SIZE, HIGH_OPTIMAL_SIZE,

```
ESTD_OPTIMAL_EXECUTIONS,          ESTD_ONEPASS_EXECUTIONS,
ESTD_MULTIPASSES_EXECUTIONS,          ESTD_TOTAL_EXECUTIONS,
IGNORED_WORKAREAS_COUNT    from    GV$PGA_TARGET_ADVICE_HISTOGRAM    where
INST_ID = USERENV('Instance')


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
V$PQ_SESSTAT
select  STATISTIC , LAST_QUERY , SESSION_TOTAL from GV$PQ_SESSTAT where inst_id
= USERENV('Instance')

V$PQ_SLAVE
select  SLAVE_NAME , STATUS , SESSIONS , IDLE_TIME_CUR , BUSY_TIME_CUR ,
CPU_SECS_CUR , MSGS_SENT_CUR , MSGS_RCVD_CUR , IDLE_TIME_TOTAL , BUSY_TIME_TOTAL
, CPU_SECS_TOTAL , MSGS_SENT_TOTAL , MSGS_RCVD_TOTAL from GV$PQ_SLAVE where
inst_id = USERENV('Instance')

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

V$PQ_SYSSTAT
select  STATISTIC , VALUE from GV$PQ_SYSSTAT where inst_id = USERENV('Instance')

V$PQ_TQSTAT
select  DFO_NUMBER , TQ_ID , SERVER_TYPE , NUM_ROWS , BYTES , OPEN_TIME ,
AVG_LATENCY , WAITS , TIMEOUTS , PROCESS , INSTANCE from GV$PQ_TQSTAT where
inst_id = USERENV('Instance')


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
V$PROCESS
select addr, pid,spid,username,serial#,terminal,program,traceid,tracefile,
background,
latchwait,latchspin,pga_used_mem,pga_alloc_mem,pga_freeable_mem,pga_max_mem from
gv$process where inst_id = USERENV('Instance')

V$PROCESS_GROUP
SELECT INDX, NAME, PID FROM gv$process_group    WHERE INST_ID =
USERENV('INSTANCE')

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
```

V$PROCESS_MEMORY
select pid, serial#, category, allocated, used, max_allocated from
gv$process_memory where inst_id = USERENV('Instance')

V$PROCESS_MEMORY_DETAIL
select pid, serial#, category, name, heap_name, bytes, allocation_count,
heap_descriptor, parent_heap_descriptor from gv$process_memory_detail where
inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$PROCESS_MEMORY_DETAIL_PROG
select pid, serial#, status from gv$process_memory_detail_prog where inst_id =
USERENV('Instance')

V$PROPAGATION_RECEIVER
select src_queue_schema, src_queue_name, src_dbname, dst_queue_schema,
dst_queue_name, startup_time, high_water_mark,
acknowledgement, last_received_msg, total_msgs, elapsed_unpickle_time,

VIEW_NAME

VIEW_DEFINITION
----
elapsed_rule_time, elapsed_enqueue_time from gv$propagation_receiver where
inst_id = USERENV('Instance')

V$PROPAGATION_SENDER
select queue_id, queue_schema, queue_name, dst_queue_schema,
dst_queue_name, startup_time, dblink, high_water_mark, acknowledgement,
schedule_status, total_msgs, total_bytes, elapsed_dequeue_time,
elapsed_pickle_time, elapsed_propagation_time, max_num_per_win,
max_size, last_msg_latency, last_msg_enqueue_time,

VIEW_NAME

VIEW_DEFINITION
----
last_msg_propagation_time, last_lcr_latency, last_lcr_creation_time,
last_lcr_propagation_time, dst_database_name from gv$propagation_sender where
inst_id = USERENV('Instance')

V$PROXY_ARCHIVEDLOG
select RECID , STAMP , DEVICE_TYPE , HANDLE , COMMENTS , MEDIA , MEDIA_POOL ,
TAG, STATUS , DELETED, THREAD# , SEQUENCE# , RESETLOGS_CHANGE# , RESETLOGS_TIME
, FIRST_CHANGE# , FIRST_TIME , NEXT_CHANGE# , NEXT_TIME , BLOCKS , BLOCK_SIZE ,
START_TIME , COMPLETION_TIME , ELAPSED_SECONDS, RMAN_STATUS_RECID,

VIEW_NAME

VIEW_DEFINITION

RMAN_STATUS_STAMP, TERMINAL, KEEP, KEEP_UNTIL, KEEP_OPTIONS from
GV$PROXY_ARCHIVEDLOG where inst_id = USERENV('Instance')

V$PROXY_ARCHIVELOG_DETAILS
select a.*, sys.dbms_rcvman.num2displaysize(output_bytes) output_bytes_display
from (select b.session_recid session_key, b.session_recid, b.session_stamp,
a.recid copy_key, a.thread#, a.sequence#, a.resetlogs_change#,
a.resetlogs_time, a.handle, a.media, a.media_pool, a.tag, a.first_change#,
a.next_change#, a.first_time, a.next_time, (a.blocks+1)*a.block_size

VIEW_NAME

VIEW_DEFINITION
---
output_bytes, a.completion_time, a.keep, a.keep_until, a.keep_options
from v$proxy_archivedlog a, v$rman_status b, (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionkey skey from dual) c, (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionfromTimeRange fTime from dual) d,
(select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime
from dual) e where a.status = 'A' and a.rman_status_recid =
b.recid (+) and a.rman_status_stamp = b.stamp (+) and
(c.skey is null or c.skey = b.session_recid) and (d.fTime is null or
d.fTime <= b.start_time) and (e.uTime is null or e.uTime >=

VIEW_NAME

VIEW_DEFINITION
---
b.end_time))a

V$PROXY_ARCHIVELOG_SUMMARY
select nvl(num_files_backed, 0), distinct_files_backed,
min_first_change#, max_next_change#, min_first_time,
max_next_time, output_bytes,
sys.dbms_rcvman.num2displaysize(output_bytes) output_bytes_display from (select
count(*) num_files_backed, min(first_change#)min_first_change#,
max(next_change#) max_next_change#, min(first_time)min_first_time,

VIEW_NAME

VIEW_DEFINITION
---
max(next_time) max_next_time, sum((blocks+1)*block_size) output_bytes
from v$proxy_archivedlog a, v$rman_status b, (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionkey skey from dual)c, (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionfromTimeRange fTime from dual) d, (select
/*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime from
dual) e where a.status = 'A' and a.rman_status_recid = b.recid

(+) and a.rman_status_stamp = b.stamp (+) and (c.skey is
null or c.skey = b.session_recid) and (d.fTime is null or d.fTime <=
b.start_time) and (e.uTime is null or e.uTime >= b.end_time)),

VIEW_NAME

VIEW_DEFINITION

(select count(*) distinct_files_backed from (select unique thread#,
sequence#, resetlogs_change#, resetlogs_time from v$proxy_archivedlog
a, v$rman_status b, (select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionkey
skey from dual)c, (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionfromTimeRange fTime from dual) d, (select
/*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime from
dual) e where a.status = 'A' and a.rman_status_recid = b.recid
(+) and a.rman_status_stamp = b.stamp (+) and (c.skey is
null or c.skey = b.session_recid) and (d.fTime is null or d.fTime <=

VIEW_NAME

VIEW_DEFINITION
----
b.start_time) and (e.uTime is null or e.uTime >= b.end_time)))

V$PROXY_COPY_DETAILS
select a.*, sys.dbms_rcvman.num2displaysize(output_bytes) output_bytes_display
from (select b.session_recid session_key, b.session_recid, b.session_stamp,
a.recid copy_key, a.file#, a.handle, a.media, a.media_pool, a.tag,
a.creation_change#, a.creation_time, a.checkpoint_change#,
a.checkpoint_time, (a.blocks+1)*a.block_size output_bytes,
a.completion_time, a.controlfile_type, keep, keep_until, keep_options

VIEW_NAME

VIEW_DEFINITION
----
from v$proxy_datafile a, v$rman_status b , (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionkey skey from dual)c, (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionfromTimeRange fTime from dual) d,
(select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime
from dual) e where a.status = 'A' and a.rman_status_recid =
b.recid (+) and a.rman_status_stamp = b.stamp (+) and
(c.skey is null or c.skey = b.session_recid) and (d.fTime is null or
d.fTime <= b.start_time) and (e.uTime is null or e.uTime >=
b.end_time))a

VIEW_NAME

VIEW_DEFINITION

V$PROXY_COPY_SUMMARY

```
select a.*,  sys.dbms_rcvman.num2displaysize(output_bytes) output_bytes_display
from (select nvl(sum(num_times_backed),0) num_copies,      sum(distinct_copies)
distinct_copies,      min(min_checkpoint_change#) min_checkpoint_change#,
max(max_checkpoint_change#) max_checkpoint_change#,
min(min_checkpoint_time) min_checkpoint_time,      max(max_checkpoint_time)
max_checkpoint_time,      sum(output_bytes) output_bytes   from     (select
unique file#, count(*) over (partition by file#,creation_change#)
```

VIEW_NAME

VIEW_DEFINITION

```
num_times_backed,       count(distinct file#)              over (partition
by file#,creation_change#,checkpoint_change#)              distinct_copies,
min(checkpoint_change#) over (partition by file#, creation_change#)
min_checkpoint_change#,      max(checkpoint_change#) over (partition by file#,
creation_change#)              max_checkpoint_change#,
min(checkpoint_time) over (partition by file#, creation_change#)
min_checkpoint_time,      max(checkpoint_time) over (partition by file#,
creation_change#)              max_checkpoint_time,
sum((blocks+1)*block_size) over (partition by file#, creation_change#)
```

VIEW_NAME

VIEW_DEFINITION

```
output_bytes     from v$proxy_datafile a, v$rman_status   b,   (select /*+
no_merge */ sys.dbms_rcvman.sv_getsessionkey skey from dual)c,   (select /*+
no_merge */ sys.dbms_rcvman.sv_getsessionfromTimeRange fTime       from dual)
d,   (select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime
from dual) e     where  a.status = 'A' and          a.rman_status_recid =
b.recid (+) and          a.rman_status_stamp = b.stamp (+) and
(c.skey is null or c.skey = b.session_recid) and          (d.fTime is null or
d.fTime <= b.start_time) and          (e.uTime is null or e.uTime >=
b.end_time)))a
```

VIEW_NAME

VIEW_DEFINITION

V$PROXY_DATAFILE

```
select  RECID , STAMP , DEVICE_TYPE , HANDLE , COMMENTS , MEDIA , MEDIA_POOL ,
TAG , STATUS , DELETED, FILE# , CREATION_CHANGE# , CREATION_TIME ,
RESETLOGS_CHANGE# , RESETLOGS_TIME , CHECKPOINT_CHANGE# , CHECKPOINT_TIME ,
ABSOLUTE_FUZZY_CHANGE# , RECOVERY_FUZZY_CHANGE# , RECOVERY_FUZZY_TIME ,
INCREMENTAL_LEVEL , ONLINE_FUZZY , BACKUP_FUZZY , BLOCKS , BLOCK_SIZE,
OLDEST_OFFLINE_RANGE, START_TIME , COMPLETION_TIME , ELAPSED_SECONDS ,
CONTROLFILE_TYPE, KEEP, KEEP_UNTIL, KEEP_OPTIONS, RMAN_STATUS_RECID,
```

VIEW_NAME

VIEW_DEFINITION

RMAN_STATUS_STAMP, FOREIGN_DBID, PLUGGED_READONLY, PLUGIN_CHANGE#,
PLUGIN_RESETLOGS_CHANGE#, PLUGIN_RESETLOGS_TIME from GV$PROXY_DATAFILE where
inst_id = USERENV('Instance')

V$PWFILE_USERS
select USERNAME , SYSDBA , SYSOPER, SYSASM from GV$PWFILE_USERS where inst_id =
USERENV('Instance')

V$PX_BUFFER_ADVICE

VIEW_NAME

VIEW_DEFINITION
----
select STATISTIC , VALUE from GV$PX_BUFFER_ADVICE where inst_id =
USERENV('Instance')

V$PX_INSTANCE_GROUP
select QC_INSTANCE_GROUP, WHY, INSTANCE_NUMBER from GV$PX_INSTANCE_GROUP where
inst_id = USERENV('Instance')

V$PX_PROCESS
select SERVER_NAME, STATUS, PID, SPID, SID, SERIAL# from GV$PX_PROCESS where

VIEW_NAME

VIEW_DEFINITION
----
inst_id = USERENV('Instance')

V$PX_PROCESS_SYSSTAT
select STATISTIC , VALUE from GV$PX_PROCESS_SYSSTAT where inst_id =
USERENV('Instance')

V$PX_SESSION
select saddr, sid, serial#, qcsid, qcserial#, qcinst_id, server_group,
server_set, server#, degree, req_degree from GV$PX_SESSION where inst_id =

VIEW_NAME

VIEW_DEFINITION

USERENV('Instance')

V$PX_SESSTAT
select saddr, sid, serial#, qcsid, qcserial#, qcinst_id, server_group,
server_set, server#, degree, req_degree, statistic#, value from GV$PX_SESSTAT
where inst_id = USERENV('Instance')

V$QUEUE

```
select  PADDR , TYPE , QUEUED , WAIT , TOTALQ from GV$QUEUE where inst_id =

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
USERENV('Instance')

V$QUEUEING_MTH
select name from gv$queueing_mth         where inst_id = userenv('instance')

V$RECOVERY_FILE_DEST
select rdi.location, rdi.slimit, (rdi.sused + rdi.scfile),
rdi.srecl+client.srecl, rdi.fcnt from x$kccrdi rdi, (select sum(recl) srecl
from (select 0 recl from dual         union         select

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
to_number(fblogreclsiz) recl         from x$krfblog         where rownum = 1
union        select sum(case when ceilasm = 1 and rlnam like '+%'
then ceil(((rlbct*rlbsz)+1)/1048576)*1048576                  else rlbct*rlbsz
end) recl         from x$kccrl, (select /*+ no_merge */ ceilasm from x$krasga)
where bitand(rlfl2, 64) = 64          and (bitand(rlfl2, 4096) = 4096 or
bitand(rlfl2, 8192) = 8192)          and rlnam is not null)) client

V$RECOVERY_FILE_STATUS
select  FILENUM , FILENAME , STATUS from GV$RECOVERY_FILE_STATUS where inst_id =

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
USERENV('Instance')

V$RECOVERY_LOG
select  THREAD# , SEQUENCE# , TIME , ARCHIVE_NAME from GV$RECOVERY_LOG where
inst_id = USERENV('Instance')

V$RECOVERY_PROGRESS
select START_TIME, TYPE, ITEM, UNITS, SOFAR, TOTAL, TIMESTAMP from
GV$RECOVERY_PROGRESS where inst_id = USERENV('Instance')

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

V$RECOVERY_STATUS
select  RECOVERY_CHECKPOINT , THREAD , SEQUENCE_NEEDED , SCN_NEEDED ,
TIME_NEEDED , PREVIOUS_LOG_NAME , PREVIOUS_LOG_STATUS , REASON from
```

GV$RECOVERY_STATUS where inst_id = USERENV('Instance')

V$RECOVER_FILE
select FILE# , "ONLINE" , ONLINE_STATUS, ERROR , CHANGE# , TIME from
GV$RECOVER_FILE where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$REDO_DEST_RESP_HISTOGRAM
select DEST_ID, TIME, DURATION, FREQUENCY from GV$REDO_DEST_RESP_HISTOGRAM
where inst_id = USERENV('Instance')

V$REPLPROP
SELECT SID, SERIAL#, NAME, DBLINK, STATE, XID, SEQUENCE FROM GV$REPLPROP WHERE
INST_ID = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$REPLQUEUE
SELECT TXNS_ENQUEUED, CALLS_ENQUEUED, TXNS_PURGED, LAST_ENQUEUE_TIME,
LAST_PURGE_TIME FROM GV$REPLQUEUE WHERE INST_ID = USERENV('Instance')

V$REQDIST
select BUCKET , COUNT from GV$REQDIST where inst_id = USERENV('Instance')

V$RESERVED_WORDS
select KEYWORD, LENGTH,   RESERVED, RES_TYPE, RES_ATTR, RES_SEMI,

VIEW_NAME

VIEW_DEFINITION

DUPLICATE   from GV$RESERVED_WORDS where inst_id = USERENV('Instance')

V$RESOURCE
select ADDR , TYPE , ID1 , ID2 from GV$RESOURCE where inst_id =
USERENV('Instance')

V$RESOURCE_LIMIT
select RESOURCE_NAME, CURRENT_UTILIZATION, MAX_UTILIZATION, INITIAL_ALLOCATION,
LIMIT_VALUE from GV$RESOURCE_LIMIT where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$RESTORE_POINT
select SCN, DATABASE_INCARNATION#, GUARANTEE_FLASHBACK_DATABASE,
STORAGE_SIZE, TIME, RESTORE_POINT_TIME, PRESERVED, NAME from
GV$RESTORE_POINT where inst_id = USERENV('Instance')

V$RESULT_CACHE_DEPENDENCY
select RESULT_ID, DEPEND_ID, OBJECT_NO from
GV$RESULT_CACHE_DEPENDENCY where inst_id=USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$RESULT_CACHE_MEMORY
select ID, CHUNK, OFFSET, FREE, OBJECT_ID,
POSITION from GV$RESULT_CACHE_MEMORY where inst_id=USERENV('Instance')

V$RESULT_CACHE_OBJECTS
select ID, TYPE, STATUS, BUCKET_NO, HASH,
NAME, NAMESPACE, CREATION_TIMESTAMP, CREATOR_UID,
DEPEND_COUNT, BLOCK_COUNT, SCN, COLUMN_COUNT,

VIEW_NAME

VIEW_DEFINITION

PIN_COUNT, SCAN_COUNT, ROW_COUNT, ROW_SIZE_MAX,
ROW_SIZE_MIN, ROW_SIZE_AVG, BUILD_TIME, LRU_NUMBER,
OBJECT_NO, INVALIDATIONS, SPACE_OVERHEAD, SPACE_UNUSED,
CACHE_ID, CACHE_KEY from GV$RESULT_CACHE_OBJECTS where
inst_id=USERENV('Instance')

V$RESULT_CACHE_STATISTICS
select ID, NAME, VALUE from GV$RESULT_CACHE_STATISTICS where
inst_id=USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$RESUMABLE
select ADDR, SID, ENABLED, STATUS, TIMEOUT, SUSPEND_TIME, RESUME_TIME, NAME,
ERROR_NUMBER, ERROR_PARAMETER1, ERROR_PARAMETER2, ERROR_PARAMETER3,
ERROR_PARAMETER4, ERROR_PARAMETER5, ERROR_MSG from GV$RESUMABLE where inst_id =
USERENV('Instance')

V$RFS_THREAD
select THREAD#, RESETLOGS_CHANGE#, RESET_TIMESTAMP, LAST_REDO_SEQ#,

VIEW_NAME

VIEW_DEFINITION

LAST_REDO_BLK#, LAST_REDO_TIME, LOW_GAP_SCN, LOW_GAP_TIME, LAST_PING_TIME FROM
gv$RFS_THREAD where INST_ID = USERENV('INSTANCE')

V$RMAN_BACKUP_JOB_DETAILS
select a.*, sys.dbms_rcvman.num2displaysize(input_bytes) input_bytes_display,
sys.dbms_rcvman.num2displaysize(output_bytes) output_bytes_display,
sys.dbms_rcvman.num2displaysize(input_bytes_per_sec)
input_bytes_per_sec_display,
sys.dbms_rcvman.num2displaysize(output_bytes_per_sec)

VIEW_NAME

VIEW_DEFINITION
----
output_bytes_per_sec_display,
sys.dbms_rcvman.sec2displaytime(elapsed_seconds) time_taken_display from (select
unique a.session_recid session_key, a.*,   decode(autobackup_count, 0, 'NO',
'YES') autobackup_done,   decode(status_weight, 2000, 'FAILED',
1900, 'RUNNING WITH ERRORS',   1500, 'RUNNING WITH
WARNINGS',   1001, 'RUNNING',
900, 'COMPLETED WITH ERRORS',   500, 'COMPLETED WITH
WARNINGS',   001, 'COMPLETED',
'FAILED') status,   decode(object_type_weight,9, 'DB FULL',

VIEW_NAME

VIEW_DEFINITION

8, 'RECVR AREA',   7, 'DB INCR',
6, 'DATAFILE FULL',   5, 'DATAFILE INCR',
4, 'ARCHIVELOG',   3, 'CONTROLFILE',
2, 'SPFILE',   1, 'BACKUPSET', null) input_type,
decode(optimized_weight, 1, 'YES', 'NO') optimized,
abs(a.end_time-a.start_time)*86400 elapsed_seconds, case when
a.input_bytes/decode(a.output_bytes,0,null, a.output_bytes) > 1 then
a.input_bytes/decode(a.output_bytes,0,null, a.output_bytes) else 1 end
compression_ratio, a.input_bytes/(decode(a.end_time-a.start_time, 0, 1,

VIEW_NAME

VIEW_DEFINITION
----
abs(a.end_time-a.start_time)*86400)) input_bytes_per_sec,
a.output_bytes/(decode(a.end_time-a.start_time, 0, 1,
abs(a.end_time-a.start_time)*86400)) output_bytes_per_sec   from (select
session_recid, session_stamp, command_id,   min(start_time) over
(partition by session_recid, session_stamp) start_time,   max(end_time)
over   (partition by session_recid, session_stamp) end_time,

sum(input_bytes) over (partition by session_recid, session_stamp)
input_bytes, sum(output_bytes) over (partition by
session_recid, session_stamp) output_bytes, max(status_weight) over

VIEW_NAME

VIEW_DEFINITION

(partition by session_recid, session_stamp)status_weight,
max(optimized_weight) over (partition by session_recid,
session_stamp) optimized_weight, max(object_type_weight) over
(partition by session_recid, session_stamp) object_type_weight,
decode(count(distinct output_device_type) over (partition by
session_recid, session_stamp),1, first_value(output_device_type)
over (partition by session_recid, session_stamp),0,
null, '*') output_device_type, sum(autobackup_count) over
(partition by session_recid, session_stamp) autobackup_count,

VIEW_NAME

VIEW_DEFINITION
----
backed_by_osb from V$RMAN_BACKUP_SUBJOB_DETAILS) a)a

V$RMAN_BACKUP_SUBJOB_DETAILS
select a.session_recid session_key, a.*, decode(nvl(b.autocnt,0), 0, 'NO',
'YES') autobackup_done, decode(status_weight, 2000, 'FAILED',
1900, 'RUNNING WITH ERRORS', 1500, 'RUNNING WITH
WARNINGS', 1001, 'RUNNING',
900, 'COMPLETED WITH ERRORS', 500, 'COMPLETED WITH
WARNINGS', 001, 'COMPLETED',

VIEW_NAME

VIEW_DEFINITION

'FAILED') status, decode(object_type_weight,9, 'DB FULL',
8, 'RECVR AREA', 7, 'DB INCR',
6, 'DATAFILE FULL', 5, 'DATAFILE INCR',
4, 'ARCHIVELOG', 3, 'CONTROLFILE',
2, 'SPFILE', 1, 'BACKUPSET', null) object_type,
decode(optimized_weight, 1, 'YES', 'NO') optimized, nvl(b.autocnt,0)
autobackup_count, case when input_bytes/decode(output_bytes,0,null,
output_bytes) > 1 then input_bytes/decode(output_bytes,0,null,
output_bytes) else 1 end compression_ratio,

VIEW_NAME

VIEW_DEFINITION
----
sys.dbms_rcvman.num2displaysize(input_bytes) input_bytes_display,
sys.dbms_rcvman.num2displaysize(output_bytes) output_bytes_display from (

```
select unique session_recid, session_stamp, operation, command_id,
min(start_time) over          (partition by session_recid, session_stamp,
operation) start_time,       max(end_time) over          (partition by
session_recid, session_stamp, operation) end_time,       sum(input_bytes) over
(partition by session_recid, session_stamp, operation) input_bytes,
sum(output_bytes) over          (partition by session_recid, session_stamp,
operation) output_bytes,       max(status_weight) over          (partition by

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
session_recid, session_stamp, operation)status_weight,
max(object_type_weight) over          (partition by session_recid,
session_stamp, operation)          object_type_weight,
max(optimized_weight) over          (partition by session_recid,
session_stamp, operation)          optimized_weight,
decode(count(distinct output_device_type) over               (partition by
session_recid, session_stamp, operation),1,
first_value(output_device_type) over               (partition by
session_recid, session_stamp, operation),0,           null, '*')

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
output_device_type,       decode(count(distinct osb_allocated) over
(partition by session_recid, session_stamp, operation),1,
first_value(osb_allocated) over               (partition by session_recid,
session_stamp, operation),0,           'NO', '*') backed_by_osb       from
(select d.*,              decode(status, 'RUNNING', 1001,
'RUNNING WITH WARNINGS', 1500,               'RUNNING WITH ERRORS', 1900,
'COMPLETED', 0001,            'COMPLETED WITH WARNINGS', 500,
'COMPLETED WITH ERRORS', 900,                        'FAILED', 2000,
2000) status_weight,           decode(object_type, 'DB FULL', 9,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
'RECVR AREA', 8,                        'DB INCR', 7,
'DATAFILE FULL', 6,                     'DATAFILE INCR', 5,
'ARCHIVELOG', 4,                        'CONTROLFILE', 3,
'SPFILE', 2,                         'BACKUPSET', 1, 0)
object_type_weight,          decode(optimized,'YES', 1, 0) optimized_weight
from        v$rman_status d        where operation like 'BACKUP%' and
row_level=1)) a,  ( select session_recid, session_stamp, count(*) autocnt from
v$rman_status     where operation like '%AUTOBACKUP%' and row_level > 1
group by session_recid, session_stamp ) b  where a.session_recid=b.session_recid

VIEW_NAME
------------------------------
```

VIEW_DEFINITION

(

V$RMAN_BACKUP_TYPE
select 9, 'DB FULL' from dual union select 8, 'RECVR AREA' from dual
union select 7, 'DB INCR' from dual union select 6, 'DATAFILE FULL'
from dual union select 5, 'DATAFILE INCR' from dual union select 4,
'ARCHIVELOG' from dual union select 3, 'CONTROLFILE' from dual union
select 2, 'SPFILE' from dual union select 1, 'BACKUPSET' from dual

VIEW_NAME

VIEW_DEFINITION

V$RMAN_COMPRESSION_ALGORITHM
SELECT algorithm_id, algorithm_name, algorithm_description,
algorithm_compatibility, database_compatibility, is_valid, is_default FROM
gv$rman_compression_algorithm WHERE inst_id = USERENV('Instance')

V$RMAN_CONFIGURATION
select CONF#, NAME, VALUE from GV$RMAN_CONFIGURATION where inst_id =
USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$RMAN_ENCRYPTION_ALGORITHMS
select algorithm_id, algorithm_name, algorithm_description, is_default,
restore_only from gv$rman_encryption_algorithms where inst_id =
USERENV('Instance')

V$RMAN_OUTPUT
select SID,RECID,STAMP,SESSION_RECID,SESSION_STAMP,OUTPUT,RMAN_STATUS_RECID,
RMAN_STATUS_STAMP, SESSION_RECID from GV$RMAN_OUTPUT where inst_id =
USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$RMAN_STATUS
SELECT
nvl(R1.SID,0),nvl(R1.RECID,R2.RSRRID),nvl(R1.STAMP,R2.RSRTST),decode(nvl(R1.ROW_
LEVEL, R2.RSRLV), 0, to_number(NULL),nvl(R1.PARENT_RECID,
RSRPI)),decode(nvl(R1.ROW_LEVEL, R2.RSRLV), 0,
to_number(NULL),nvl(R1.PARENT_STAMP, R2.RSRPS)),decode (nvl(R1.ROW_LEVEL,
R2.RSRLV), 0, nvl(R1.RECID,R2.RSRRID), R2.RSR0I), decode (nvl(R1.ROW_LEVEL,

R2.RSRLV), 0, nvl(R1.STAMP,R2.RSRTST), R2.RSR0S), nvl(R1.ROW_LEVEL,

VIEW_NAME

VIEW_DEFINITION

R2.RSRLV),nvl(R1.ROW_TYPE, decode(R2.RSRLV, 0, 'SESSION',1, 'COMMAND','RECURSIVE
OPERATION')),nvl(R1.COMMAND_ID,R2.RSRCI), nvl(R1.OPERATION,
UPPER(R2.RSROP)),nvl(R1.STATUS, decode(bitand(R2.RSRIS,2+1), 2,decode(R2.RSRES,
1, 'RUNNING',1+8, 'RUNNING WITH WARNINGS', 1+16, 'RUNNING WITH ERRORS', 1+8+16,
'RUNNING WITH ERRORS', 2, 'COMPLETED', 2+8, 'COMPLETED WITH WARNINGS', 2+16,
'COMPLETED WITH ERRORS', 2+8+16, 'COMPLETED WITH ERRORS', 'FAILED'), 'FAILED')),
decode(R2.RSRMP, 0, nvl(HH.MBYTES, 0),R2.RSRMP), nvl(R1.START_TIME,
to_date(R2.RSRST,'MM/DD/RR HH24:MI:SS', 'NLS_CALENDAR=Gregorian')),
nvl(R1.END_TIME, to_date(R2.RSRET,'MM/DD/RR

VIEW_NAME

VIEW_DEFINITION

HH24:MI:SS','NLS_CALENDAR=Gregorian')), decode(R2.RSRIM, 0, nvl(HH.INPBYTES, 0),
decode(bitand(R2.RSRIM, 2147483648), 0,R2.RSRIM, 2147483648,
bitand(R2.RSRIM,2147483647)*1024*1024)), decode(R2.RSROM, 0, nvl(HH.OUTBYTES,
0), decode(bitand(R2.RSROM, 2147483648), 0,R2.RSROM, 2147483648,
bitand(R2.RSROM,2147483647)*1024*1024)), decode(bitand(R2.RSRFL, 32), 1, 'YES',
'NO'), nvl(nvl(nvl(nvl(nvl(nvl(decode(bitand(R2.RSRFL, 1+128),1, 'DB
FULL',null), decode(bitand(R2.RSRFL, 64), 64, 'RECVR AREA', null)),
decode(bitand(R2.RSRFL, 1+128), 129, 'DB INCR', null)), decode(bitand(R2.RSRFL,
2+128),2, 'DATAFILE FULL', 130, 'DATAFILE INCR', null)), decode(bitand(R2.RSRFL,

VIEW_NAME

VIEW_DEFINITION
---
4), 4, 'ARCHIVELOG', null)), decode(bitand(R2.RSRFL, 8), 8, 'CONTROLFILE',
null)), decode(bitand(R2.RSRFL, 16), 16, 'SPFILE', null)),
decode(bitand(R2.RSRFL, 256), 256, 'BACKUPSET', null)), decode(bitand(R2.RSRIS,
32+16+8+4), 0, odev.device_type,   4, 'DISK', 8, 'SBT_TAPE', 16, '*', null)
device_type, decode(bitand(R2.RSRFL, 512), 512, 'YES', 'NO') OSB FROM X$KCCRSR
R2, GV$RMAN_STATUS_CURRENT R1, (SELECT R.RSRRID RECID, R.RSRTST STAMP,
sum(aggrcol)/(1024*1024) MBYTES,   sum(inpcol) INPBYTES, sum(outcol) OUTBYTES
from x$kccrsr R,   (select rman_status_recid, rman_status_stamp,   sum(case
when type=3 then blocks*block_size else 0 end) aggrcol,   sum(case when

VIEW_NAME

VIEW_DEFINITION
---
type=1 then blocks*block_size else 0 end) inpcol,   sum(case when type=2 then
blocks*block_size else 0 end) outcol   from x$ksfqp group by
rman_status_recid, rman_status_stamp) RS   where R.RSRRID   =
RS.RMAN_STATUS_RECID(+)   and R.RSRTST   = RS.RMAN_STATUS_STAMP(+)   group by

R.RSRRID, R.RSRTST) HH, (SELECT unique R.RSRRID RECID, R.RSRTST STAMP,
device_type from x$kccrsr R, (SELECT RMAN_STATUS_RECID, RMAN_STATUS_STAMP,
decode(count(distinct devtype) over (partition by RMAN_STATUS_RECID,
RMAN_STATUS_STAMP),1, first_value(devtype) over (partition by
RMAN_STATUS_RECID, RMAN_STATUS_STAMP), 0, null,'*')

VIEW_NAME

VIEW_DEFINITION
---
device_type from x$ksfqp where 2 = TYPE) RS where R.RSRRID =
RS.RMAN_STATUS_RECID(+) and R.RSRTST = RS.RMAN_STATUS_STAMP(+)) ODEV WHERE
nvl(R1.RECID,R2.RSRRID) = HH.RECID AND nvl(R1.STAMP,R2.RSRTST) = HH.STAMP AND
nvl(R1.RECID,R2.RSRRID) = ODEV.RECID AND nvl(R1.STAMP,R2.RSRTST) = ODEV.STAMP
AND R2.RSRRID = R1.RECID(+) AND R2.RSRTST = R1.STAMP(+)

V$ROLLSTAT
select USN , LATCH, EXTENTS , RSSIZE , WRITES , XACTS , GETS , WAITS , OPTSIZE
, HWMSIZE , SHRINKS , WRAPS , EXTENDS , AVESHRINK , AVEACTIVE , STATUS , CUREXT

VIEW_NAME

VIEW_DEFINITION

, CURBLK from GV$ROLLSTAT where inst_id = USERENV('Instance')

V$ROWCACHE
select cache#,type,subordinate#,parameter,count,usage,fixed,
gets,getmisses,scans,scanmisses,scancompletes,modifications,flushes,dlm_requests
,dlm_conflicts,dlm_releases from gv$rowcache where inst_id = USERENV('Instance')

V$ROWCACHE_PARENT

VIEW_NAME

VIEW_DEFINITION
---
select indx, hash, address, cache#, cache_name, existent, lock_mode,
lock_request, txn, saddr, inst_lock_request, inst_lock_release, inst_lock_type,
inst_lock_id1, inst_lock_id2, key from gv$rowcache_parent where inst_id =
USERENV('Instance')

V$ROWCACHE_SUBORDINATE
select indx, hash, address, cache#, subcache#, subcache_name, existent, parent,
key from gv$rowcache_subordinate where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$RSRCMGRMETRIC
SELECT begin_time, end_time, intsize_csec, sequence#,
consumer_group_id, consumer_group_name, cpu_consumed_time,
cpu_wait_time, io_requests, io_megabytes FROM
gv$rsrcmgrmetric WHERE inst_id = USERENV('INSTANCE')

V$RSRCMGRMETRIC_HISTORY
SELECT begin_time, end_time, intsize_csec, sequence#,
consumer_group_id, consumer_group_name, cpu_consumed_time,

VIEW_NAME

VIEW_DEFINITION
---
cpu_wait_time, io_requests, io_megabytes FROM
gv$rsrcmgrmetric_history WHERE inst_id = USERENV('INSTANCE')

V$RSRC_CONSUMER_GROUP
select id, name, active_sessions, execution_waiters, requests,
cpu_wait_time, cpu_waits, consumed_cpu_time, yields, queue_length,
current_undo_consumption, active_session_limit_hit, undo_limit_hit,
switches_in_cpu_time, switches_out_cpu_time, switches_in_io_megabytes,
switches_out_io_megabytes, switches_in_io_requests,

VIEW_NAME

VIEW_DEFINITION
---
switches_out_io_requests, sql_canceled, active_sessions_killed,
idle_sessions_killed, idle_blkr_sessions_killed, queued_time,
queue_time_outs, io_service_time, io_service_waits,
small_read_megabytes, small_write_megabytes, large_read_megabytes,
large_write_megabytes, small_read_requests, small_write_requests,
large_read_requests, large_write_requests from gv$rsrc_consumer_group
where inst_id = userenv('instance')

V$RSRC_CONSUMER_GROUP_CPU_MTH

VIEW_NAME

VIEW_DEFINITION
---
select name from gv$rsrc_consumer_group_cpu_mth where inst_id =
userenv('instance')

V$RSRC_CONS_GROUP_HISTORY
select sequence#, id, name, requests, cpu_wait_time, cpu_waits,
consumed_cpu_time, yields, active_sess_limit_hit, undo_limit_hit,
switches_in_cpu_time, switches_out_cpu_time, switches_in_io_megabytes,
switches_out_io_megabytes, switches_in_io_requests,
switches_out_io_requests, sql_canceled, active_sess_killed,

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
idle_sess_killed, idle_blkr_sess_killed,         queued_time, queue_time_outs,
io_service_time, io_service_waits,         small_read_megabytes,
small_write_megabytes,         large_read_megabytes, large_write_megabytes,
small_read_requests, small_write_requests,         large_read_requests,
large_write_requests         from gv$rsrc_cons_group_history         where
inst_id = userenv('instance')

V$RSRC_PLAN
select id, name, is_top_plan, cpu_managed         from gv$rsrc_plan

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
where inst_id = userenv('instance')

V$RSRC_PLAN_CPU_MTH
select name from gv$rsrc_plan_cpu_mth         where inst_id =
userenv('instance')

V$RSRC_PLAN_HISTORY
select sequence#, id, name,         start_time, end_time,
enabled_by_scheduler, window_name, allowed_automated_switches,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
cpu_managed         from gv$rsrc_plan_history         where inst_id =
userenv('instance')

V$RSRC_SESSION_INFO
select  sid, current_consumer_group_id, orig_consumer_group_id,
mapping_attribute, mapped_consumer_group,         state, active,
current_idle_time,         current_cpu_wait_time, cpu_wait_time,
current_cpu_waits, cpu_waits,         current_consumed_cpu_time,
consumed_cpu_time,         current_active_time,         active_time,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
current_queued_time, queued_time, current_yields,         yields,
current_undo_consumption, max_undo_consumption,         sql_canceled,
queue_time_outs, estimated_execution_limit_hit,
current_io_service_time, io_service_time,         current_io_service_waits,
io_service_waits,         current_small_read_megabytes, small_read_megabytes,
current_large_read_megabytes, large_read_megabytes,
```

current_small_write_megabytes, small_write_megabytes,
current_large_write_megabytes, large_write_megabytes,
current_small_read_requests, small_read_requests,

VIEW_NAME

VIEW_DEFINITION
----
current_small_write_requests, small_write_requests,
current_large_read_requests, large_read_requests,
current_large_write_requests, large_write_requests from
gv$rsrc_session_info where inst_id = userenv('instance')

V$RULE
select RULE_SET_OBJECT_ID, EVALUATION_CONTEXT_OBJECT_ID, RULE_OWNER, RULE_NAME,
RULE_CONDITION, TRUE_HITS, MAYBE_HITS, SQL_EVALUATIONS from GV$RULE where
inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$RULE_SET
select OWNER, NAME, CPU_TIME, ELAPSED_TIME, FIRST_LOAD_TIME, LAST_LOAD_TIME,
LAST_LOADING_TIME, SHARABLE_MEM, RELOADS, INVALIDATIONS, EVALUATIONS,
FIRST_HIT_EVALUATIONS, SIMPLE_RULES_ONLY_EVALUATIONS, SQL_FREE_EVALUATIONS,
SQL_EXECUTIONS, CONDITIONS_PROCESSED, TRUE_RULES, MAYBE_RULES,
VARIABLE_VALUE_FUNCTION_CALLS, VARIABLE_METHOD_FUNCTION_CALLS,
EVALUATION_FUNCTION_CALLS from GV$RULE_SET where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$RULE_SET_AGGREGATE_STATS
select NAME, VALUE from GV$RULE_SET_AGGREGATE_STATS where inst_id =
USERENV('Instance')

V$SCHEDULER_RUNNING_JOBS
select session_id, session_serial_num, job_id, paddr, os_process_id,
session_stat_cpu from gv$scheduler_running_jobs where inst_id =
USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$SECUREFILE_TIMER
SELECT NAME, LAYER_ID, OWNTIME, MAXTIME, MINTIME, INVOCATIONS, LAYER_NAME FROM

GV$SECUREFILE_TIMER

V$SEGMENT_STATISTICS

```
select owner,                                   object_name,
subobject_name,                                 tablespace_name,
ts#,                                          obj#,
dataobj#,                                       object_type,
```

VIEW_NAME

VIEW_DEFINITION

```
statistic_name,                                 statistic#,
value                                       from  gv$segment_statistics
where inst_id = userenv('instance')
```

V$SEGSTAT

```
select ts#,                                         obj#,
dataobj#,                                       statistic_name,
statistic#,                                     value
from  gv$segstat where inst_id = userenv('instance')
```

VIEW_NAME

VIEW_DEFINITION

V$SEGSTAT_NAME

```
select statistic#,                                  name,
sampled                                     from gv$segstat_name
where inst_id = userenv('instance')
```

V$SERVICEMETRIC

```
SELECT begin_time, end_time, intsize_csec,          group_id,
service_name_hash, service_name,            ctmhash, elapsedpercall, cpupercall,
```

VIEW_NAME

VIEW_DEFINITION

```
dbtimepercall,          callspersec, dbtimepersec, goodness, delta, flags
FROM   gv$servicemetric         WHERE  inst_id = USERENV('INSTANCE')
```

V$SERVICEMETRIC_HISTORY

```
SELECT begin_time, end_time, intsize_csec,          group_id,
service_name_hash, service_name,            ctmhash, elapsedpercall, cpupercall,
dbtimepercall,          callspersec, dbtimepersec         FROM
gv$servicemetric_history         WHERE  inst_id = USERENV('INSTANCE')
```

VIEW_NAME

VIEW_DEFINITION

V$SERVICES
select SERVICE_ID, NAME, NAME_HASH, NETWORK_NAME, CREATION_DATE,
CREATION_DATE_HASH, GOAL, DTP, AQ_HA_NOTIFICATION, CLB_GOAL from GV$SERVICES
where inst_id = USERENV('Instance')

V$SERVICE_EVENT
select service_name, service_name_hash,event,event_id, total_waits,
total_timeouts, time_waited,average_wait, max_wait,time_waited_micro from
gv$service_event where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$SERVICE_STATS
select SERVICE_NAME_HASH , SERVICE_NAME , STAT_ID , STAT_NAME , VALUE from
GV$SERVICE_STATS where inst_id = USERENV('Instance')

V$SERVICE_WAIT_CLASS
select service_name, service_name_hash,wait_class_id, wait_class#,
wait_class,total_waits, time_waited from gv$service_wait_class where inst_id =
USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$SERV_MOD_ACT_STATS
select AGGREGATION_TYPE, SERVICE_NAME, MODULE, ACTION, STAT_ID,   STAT_NAME,
VALUE from GV$SERV_MOD_ACT_STATS where inst_id = USERENV('Instance')

V$SESSION
select SADDR , SID , SERIAL# , AUDSID , PADDR , USER# , USERNAME , COMMAND ,
OWNERID, TADDR , LOCKWAIT , STATUS , SERVER , SCHEMA# , SCHEMANAME ,OSUSER ,
PROCESS , MACHINE , TERMINAL , PROGRAM , TYPE , SQL_ADDRESS , SQL_HASH_VALUE,

VIEW_NAME

VIEW_DEFINITION

SQL_ID, SQL_CHILD_NUMBER , SQL_EXEC_START, SQL_EXEC_ID, PREV_SQL_ADDR ,
PREV_HASH_VALUE , PREV_SQL_ID, PREV_CHILD_NUMBER , PREV_EXEC_START ,
PREV_EXEC_ID , PLSQL_ENTRY_OBJECT_ID, PLSQL_ENTRY_SUBPROGRAM_ID,
PLSQL_OBJECT_ID, PLSQL_SUBPROGRAM_ID, MODULE , MODULE_HASH , ACTION ,
ACTION_HASH , CLIENT_INFO , FIXED_TABLE_SEQUENCE , ROW_WAIT_OBJ# ,
ROW_WAIT_FILE# , ROW_WAIT_BLOCK# , ROW_WAIT_ROW# , LOGON_TIME , LAST_CALL_ET ,
PDML_ENABLED , FAILOVER_TYPE , FAILOVER_METHOD , FAILED_OVER,
RESOURCE_CONSUMER_GROUP, PDML_STATUS, PDDL_STATUS, PQ_STATUS,

CURRENT_QUEUE_DURATION, CLIENT_IDENTIFIER, BLOCKING_SESSION_STATUS,

VIEW_NAME

VIEW_DEFINITION

BLOCKING_INSTANCE,BLOCKING_SESSION,SEQ#,
EVENT#,EVENT,P1TEXT,P1,P1RAW,P2TEXT,P2,P2RAW, P3TEXT,P3,P3RAW,WAIT_CLASS_ID,
WAIT_CLASS#,WAIT_CLASS,WAIT_TIME,
SECONDS_IN_WAIT,STATE,WAIT_TIME_MICRO,TIME_REMAINING_MICRO,
TIME_SINCE_LAST_WAIT_MICRO,SERVICE_NAME, SQL_TRACE, SQL_TRACE_WAITS,
SQL_TRACE_BINDS, SQL_TRACE_PLAN_STATS, SESSION_EDITION_ID, CREATOR_ADDR,
CREATOR_SERIAL# from GV$SESSION where inst_id = USERENV('Instance')

V$SESSION_CONNECT_INFO

VIEW_NAME

VIEW_DEFINITION
---
select sid,serial#,authentication_type,osuser,network_service_banner,
client_charset,client_connection,client_oci_library,client_version,
client_driver, client_lobattr, client_regid from gv$session_connect_info where
inst_id = USERENV('Instance')

V$SESSION_CURSOR_CACHE
select MAXIMUM , COUNT , OPENS , HITS , HIT_RATIO from GV$SESSION_CURSOR_CACHE
where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$SESSION_EVENT
select sid,event,total_waits,total_timeouts,time_waited,average_wait,
max_wait,time_waited_micro, event_id, wait_class_id,wait_class#, wait_class
from gv$session_event where inst_id = USERENV('Instance')

V$SESSION_FIX_CONTROL
select SESSION_ID, BUGNO, VALUE, SQL_FEATURE,
DESCRIPTION, OPTIMIZER_FEATURE_ENABLE, EVENT, IS_DEFAULT
from GV$SESSION_FIX_CONTROL where inst_id=USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$SESSION_LONGOPS
select SID, SERIAL#, OPNAME, TARGET, TARGET_DESC,
SOFAR, TOTALWORK, UNITS, START_TIME, LAST_UPDATE_TIME,

TIMESTAMP, TIME_REMAINING, ELAPSED_SECONDS, CONTEXT, MESSAGE,
USERNAME, SQL_ADDRESS, SQL_HASH_VALUE, SQL_ID,
SQL_PLAN_HASH_VALUE, SQL_EXEC_START, SQL_EXEC_ID,
SQL_PLAN_LINE_ID, SQL_PLAN_OPERATION, SQL_PLAN_OPTIONS, QCSID from
GV$SESSION_LONGOPS where

VIEW_NAME

VIEW_DEFINITION
---
inst_id = USERENV('Instance')

V$SESSION_OBJECT_CACHE
select
pins,hits,true_hits,hit_ratio,true_hit_ratio,object_refreshes,cache_refreshes,ob
ject_flushes,cache_flushes,cache_shrinks,cached_objects,pinned_objects,cache_siz
e,optimal_size,maximum_size from gv$session_object_cache where
inst_id=userenv('Instance')

VIEW_NAME

VIEW_DEFINITION

V$SESSION_WAIT
select sid,seq#,event,p1text,p1,p1raw,p2text,p2,p2raw,p3text,
p3,p3raw,wait_class_id, wait_class#,wait_class,wait_time,seconds_in_wait,
state,wait_time_micro,time_remaining_micro,time_since_last_wait_micro from
gv$session_wait where inst_id = USERENV('Instance')

V$SESSION_WAIT_CLASS
select sid,serial#,wait_class_id, wait_class#,wait_class,total_waits,
time_waited from gv$session_wait_class where inst_id = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$SESSION_WAIT_HISTORY
select sid,seq#,event#,event,p1text,p1,p2text,p2,p3text,p3,wait_time,
wait_time_micro,time_since_last_wait_micro from gv$session_wait_history where
inst_id = USERENV('Instance')

V$SESSMETRIC
SELECT begin_time, end_time, intsize_csec, session_id, serial_num,
cpu, physical_reads, logical_reads, pga_memory, hard_parses,

VIEW_NAME

VIEW_DEFINITION

```
soft_parses,          physical_read_pct, logical_read_pct         FROM
gv$sessmetric         WHERE  inst_id = USERENV('INSTANCE')

V$SESSTAT
select  SID , STATISTIC# , VALUE from GV$SESSTAT where inst_id =
USERENV('Instance')

V$SESS_IO
select  SID , BLOCK_GETS , CONSISTENT_GETS , PHYSICAL_READS , BLOCK_CHANGES ,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
CONSISTENT_CHANGES from GV$SESS_IO where inst_id = USERENV('Instance')

V$SESS_TIME_MODEL
select SID, STAT_ID, STAT_NAME, VALUE from GV$SESS_TIME_MODEL  where inst_id =
USERENV('Instance')

V$SES_OPTIMIZER_ENV
select SID,                    ID,                    NAME,
SQL_FEATURE,              ISDEFAULT,                VALUE

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
from   GV$SES_OPTIMIZER_ENV     where  INST_ID = USERENV('Instance')

V$SGA
select  NAME , VALUE from GV$SGA where inst_id = USERENV('Instance')

V$SGAINFO
select name, bytes, resizeable from gv$sgainfo  where inst_id =
USERENV('Instance')


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
V$SGASTAT
select  POOL, NAME , BYTES from GV$SGASTAT where inst_id = USERENV('Instance')

V$SGA_CURRENT_RESIZE_OPS
select component, oper_type, oper_mode, parameter, initial_size,
target_size, current_size, start_time, last_update_time   from
gv$sga_current_resize_ops   where inst_id = USERENV('Instance')

V$SGA_DYNAMIC_COMPONENTS
```

VIEW_NAME

VIEW_DEFINITION
---
select component, current_size, min_size, max_size, user_specified_size,
oper_count, last_oper_type, last_oper_mode, last_oper_time, granule_size from
gv$sga_dynamic_components where inst_id = USERENV('Instance')

V$SGA_DYNAMIC_FREE_MEMORY
select current_size from gv$sga_dynamic_free_memory where inst_id =
USERENV('Instance')

V$SGA_RESIZE_OPS

VIEW_NAME

VIEW_DEFINITION
---
select component, oper_type, oper_mode, parameter, initial_size,
target_size, final_size, status, start_time, end_time from gv$sga_resize_ops
where inst_id = USERENV('Instance')

V$SGA_TARGET_ADVICE
select sga_size, sga_size_factor, estd_db_time,
estd_db_time_factor, estd_physical_reads from
gv$sga_target_advice where inst_id = userenv('instance')

VIEW_NAME

VIEW_DEFINITION

V$SHARED_POOL_ADVICE
select shared_pool_size_for_estimate, shared_pool_size_factor, estd_lc_size,
estd_lc_memory_objects, estd_lc_time_saved, estd_lc_time_saved_factor,
estd_lc_load_time, estd_lc_load_time_factor, estd_lc_memory_object_hits from
gv$shared_pool_advice where inst_id = USERENV('Instance')

V$SHARED_POOL_RESERVED
select FREE_SPACE , AVG_FREE_SIZE , FREE_COUNT , MAX_FREE_SIZE , USED_SPACE ,
AVG_USED_SIZE , USED_COUNT , MAX_USED_SIZE , REQUESTS , REQUEST_MISSES ,

VIEW_NAME

VIEW_DEFINITION

LAST_MISS_SIZE , MAX_MISS_SIZE , REQUEST_FAILURES , LAST_FAILURE_SIZE ,
ABORTED_REQUEST_THRESHOLD , ABORTED_REQUESTS , LAST_ABORTED_SIZE from
GV$SHARED_POOL_RESERVED where inst_id = USERENV('Instance')

V$SHARED_SERVER
select NAME , PADDR , STATUS , MESSAGES , BYTES , BREAKS , CIRCUIT , IDLE ,

BUSY , REQUESTS from GV$SHARED_SERVER where inst_id = USERENV('Instance')

V$SHARED_SERVER_MONITOR

VIEW_NAME

VIEW_DEFINITION
---
select MAXIMUM_CONNECTIONS , MAXIMUM_SESSIONS , SERVERS_STARTED , SERVERS_TERMINATED , SERVERS_HIGHWATER from GV$SHARED_SERVER_MONITOR where inst_id = USERENV('Instance')

V$SORT_SEGMENT
select TABLESPACE_NAME , SEGMENT_FILE , SEGMENT_BLOCK , EXTENT_SIZE , CURRENT_USERS , TOTAL_EXTENTS , TOTAL_BLOCKS , USED_EXTENTS , USED_BLOCKS , FREE_EXTENTS , FREE_BLOCKS , ADDED_EXTENTS , EXTENT_HITS , FREED_EXTENTS , FREE_REQUESTS , MAX_SIZE , MAX_BLOCKS , MAX_USED_SIZE , MAX_USED_BLOCKS ,

VIEW_NAME

VIEW_DEFINITION

MAX_SORT_SIZE , MAX_SORT_BLOCKS , RELATIVE_FNO from GV$SORT_SEGMENT where inst_id = USERENV('Instance')

V$SORT_USAGE
select USERNAME , "USER" , SESSION_ADDR , SESSION_NUM , SQLADDR , SQLHASH, SQL_ID, TABLESPACE , CONTENTS , SEGTYPE , SEGFILE# , SEGBLK# ,EXTENTS , BLOCKS , SEGRFNO# from GV$SORT_USAGE where inst_id = USERENV('Instance')

V$SPPARAMETER

VIEW_NAME

VIEW_DEFINITION
---
select SID, NAME, TYPE, VALUE, DISPLAY_VALUE, ISSPECIFIED, ORDINAL, UPDATE_COMMENT from GV$SPPARAMETER where INST_id = USERENV('Instance')

V$SQL
select SQL_TEXT , SQL_FULLTEXT , SQL_ID, SHARABLE_MEM , PERSISTENT_MEM , RUNTIME_MEM , SORTS , LOADED_VERSIONS , OPEN_VERSIONS , USERS_OPENING , FETCHES , EXECUTIONS , PX_SERVERS_EXECUTIONS , END_OF_FETCH_COUNT, USERS_EXECUTING , LOADS , FIRST_LOAD_TIME, INVALIDATIONS, PARSE_CALLS , DISK_READS , DIRECT_WRITES , BUFFER_GETS , APPLICATION_WAIT_TIME, CONCURRENCY_WAIT_TIME, CLUSTER_WAIT_TIME,

VIEW_NAME

VIEW_DEFINITION

USER_IO_WAIT_TIME, PLSQL_EXEC_TIME, JAVA_EXEC_TIME, ROWS_PROCESSED , COMMAND_TYPE , OPTIMIZER_MODE , OPTIMIZER_COST, OPTIMIZER_ENV,

OPTIMIZER_ENV_HASH_VALUE, PARSING_USER_ID , PARSING_SCHEMA_ID ,
PARSING_SCHEMA_NAME, KEPT_VERSIONS , ADDRESS , TYPE_CHK_HEAP , HASH_VALUE,
OLD_HASH_VALUE, PLAN_HASH_VALUE, CHILD_NUMBER, SERVICE, SERVICE_HASH, MODULE,
MODULE_HASH , ACTION , ACTION_HASH , SERIALIZABLE_ABORTS , OUTLINE_CATEGORY,
CPU_TIME, ELAPSED_TIME, OUTLINE_SID, CHILD_ADDRESS, SQLTYPE, REMOTE,
OBJECT_STATUS, LITERAL_HASH_VALUE, LAST_LOAD_TIME, IS_OBSOLETE,
IS_BIND_SENSITIVE, IS_BIND_AWARE, IS_SHAREABLE,CHILD_LATCH, SQL_PROFILE,

VIEW_NAME

VIEW_DEFINITION

SQL_PATCH, SQL_PLAN_BASELINE, PROGRAM_ID, PROGRAM_LINE#,
EXACT_MATCHING_SIGNATURE, FORCE_MATCHING_SIGNATURE, LAST_ACTIVE_TIME,
BIND_DATA,
TYPECHECK_MEM from GV$SQL where inst_id = USERENV('Instance')

V$SQLAREA
select SQL_TEXT, SQL_FULLTEXT, SQL_ID,
SHARABLE_MEM, PERSISTENT_MEM, RUNTIME_MEM, SORTS,
VERSION_COUNT, LOADED_VERSIONS, OPEN_VERSIONS,
USERS_OPENING, FETCHES, EXECUTIONS,

VIEW_NAME

VIEW_DEFINITION

PX_SERVERS_EXECUTIONS, END_OF_FETCH_COUNT, USERS_EXECUTING,
LOADS, FIRST_LOAD_TIME, INVALIDATIONS,
PARSE_CALLS, DISK_READS, DIRECT_WRITES,
BUFFER_GETS, APPLICATION_WAIT_TIME, CONCURRENCY_WAIT_TIME,
CLUSTER_WAIT_TIME, USER_IO_WAIT_TIME, PLSQL_EXEC_TIME,
JAVA_EXEC_TIME, ROWS_PROCESSED, COMMAND_TYPE,
OPTIMIZER_MODE, OPTIMIZER_COST, OPTIMIZER_ENV,
OPTIMIZER_ENV_HASH_VALUE, PARSING_USER_ID,
PARSING_SCHEMA_ID, PARSING_SCHEMA_NAME, KEPT_VERSIONS,

VIEW_NAME

VIEW_DEFINITION

ADDRESS, HASH_VALUE, OLD_HASH_VALUE,
PLAN_HASH_VALUE, MODULE, MODULE_HASH, ACTION,
ACTION_HASH, SERIALIZABLE_ABORTS, OUTLINE_CATEGORY,
CPU_TIME, ELAPSED_TIME, OUTLINE_SID,
LAST_ACTIVE_CHILD_ADDRESS, REMOTE, OBJECT_STATUS,
LITERAL_HASH_VALUE, LAST_LOAD_TIME, IS_OBSOLETE,
IS_BIND_SENSITIVE, IS_BIND_AWARE, IS_SHAREABLE,
CHILD_LATCH, SQL_PROFILE, SQL_PATCH,
SQL_PLAN_BASELINE, PROGRAM_ID, PROGRAM_LINE#,

VIEW_NAME

```
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
EXACT_MATCHING_SIGNATURE,          FORCE_MATCHING_SIGNATURE,
LAST_ACTIVE_TIME,          BIND_DATA,          TYPECHECK_MEM from GV$SQLAREA
where inst_id = USERENV('Instance')

V$SQLAREA_PLAN_HASH
select  SQL_TEXT,          SQL_FULLTEXT,          ADDRESS,          HASH_VALUE,
SQL_ID,          PLAN_HASH_VALUE,          VERSION_COUNT,
LAST_ACTIVE_CHILD_ADDRESS,          SHARABLE_MEM,          PERSISTENT_MEM,
RUNTIME_MEM,          SORTS,          LOADED_VERSIONS,          OPEN_VERSIONS,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
USERS_OPENING,          USERS_EXECUTING,          FETCHES,          EXECUTIONS,
PX_SERVERS_EXECUTIONS,          END_OF_FETCH_COUNT,          LOADS,
FIRST_LOAD_TIME,          LAST_LOAD_TIME,          LAST_ACTIVE_TIME,
INVALIDATIONS,          PARSE_CALLS,          DISK_READS,
DIRECT_WRITES,          BUFFER_GETS,          CPU_TIME,          ELAPSED_TIME,
APPLICATION_WAIT_TIME,          CONCURRENCY_WAIT_TIME,
CLUSTER_WAIT_TIME,          USER_IO_WAIT_TIME,          PLSQL_EXEC_TIME,
JAVA_EXEC_TIME,          ROWS_PROCESSED,          COMMAND_TYPE,
OPTIMIZER_MODE,          OPTIMIZER_COST,          OPTIMIZER_ENV,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
OPTIMIZER_ENV_HASH_VALUE,          PARSING_USER_ID,          PARSING_SCHEMA_ID,
PARSING_SCHEMA_NAME,          KEPT_VERSIONS,          MODULE,
MODULE_HASH,          ACTION,          ACTION_HASH,
SERIALIZABLE_ABORTS,          OUTLINE_CATEGORY,          OUTLINE_SID,
REMOTE,          OBJECT_STATUS,          LITERAL_HASH_VALUE,
SQL_PROFILE,          PROGRAM_ID,          PROGRAM_LINE#,
EXACT_MATCHING_SIGNATURE,          FORCE_MATCHING_SIGNATURE,          BIND_DATA,
TYPECHECK_MEM from GV$SQLAREA_PLAN_HASH where inst_id = USERENV('Instance')


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
V$SQLFN_ARG_METADATA
select FUNC_ID,          ARGNUM,          DATATYPE,
DESCR       from GV$SQLFN_ARG_METADATA where inst_id=USERENV('Instance')

V$SQLFN_METADATA
select FUNC_ID,          NAME,          MINARGS,
MAXARGS,          DATATYPE,          VERSION,          ANALYTIC,
```

AGGREGATE, DISP_TYPE, USAGE, DESCR
from GV$SQLFN_METADATA where inst_id=USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$SQLSTATS
select SQL_TEXT, SQL_FULLTEXT, SQL_ID, LAST_ACTIVE_TIME,
LAST_ACTIVE_CHILD_ADDRESS, PLAN_HASH_VALUE, PARSE_CALLS, DISK_READS,
DIRECT_WRITES, BUFFER_GETS, ROWS_PROCESSED, SERIALIZABLE_ABORTS, FETCHES,
EXECUTIONS, END_OF_FETCH_COUNT, LOADS, VERSION_COUNT, INVALIDATIONS,
PX_SERVERS_EXECUTIONS, CPU_TIME, ELAPSED_TIME, AVG_HARD_PARSE_TIME,
APPLICATION_WAIT_TIME, CONCURRENCY_WAIT_TIME, CLUSTER_WAIT_TIME,
USER_IO_WAIT_TIME, PLSQL_EXEC_TIME, JAVA_EXEC_TIME, SORTS, SHARABLE_MEM,

VIEW_NAME

VIEW_DEFINITION

TOTAL_SHARABLE_MEM, TYPECHECK_MEM FROM gv$sqlstats where
inst_id=USERENV('Instance')

V$SQLTEXT
select ADDRESS, HASH_VALUE, SQL_ID, COMMAND_TYPE , PIECE, SQL_TEXT from
GV$SQLTEXT where inst_id = USERENV('Instance')

V$SQLTEXT_WITH_NEWLINES
select ADDRESS, HASH_VALUE, SQL_ID, COMMAND_TYPE, PIECE, SQL_TEXT

VIEW_NAME

VIEW_DEFINITION
---
from GV$SQLTEXT_WITH_NEWLINES where inst_id = USERENV('Instance')

V$SQL_BIND_DATA
select CURSOR_NUM , POSITION , DATATYPE , SHARED_MAX_LEN , PRIVATE_MAX_LEN ,
ARRAY_SIZE , PRECISION , SCALE , SHARED_FLAG , SHARED_FLAG2 , BUF_ADDRESS ,
BUF_LENGTH , VAL_LENGTH , BUF_FLAG , INDICATOR , VALUE from GV$SQL_BIND_DATA
where inst_id = USERENV('Instance')

V$SQL_BIND_METADATA

VIEW_NAME

VIEW_DEFINITION
---
select ADDRESS , POSITION , DATATYPE , MAX_LENGTH , ARRAY_LEN , BIND_NAME from
GV$SQL_BIND_METADATA where inst_id = USERENV('Instance')

V$SQL_CS_HISTOGRAM
select address, hash_value, sql_id, child_number, bucket_id, count from
GV$SQL_CS_HISTOGRAM where inst_id=USERENV('Instance')

V$SQL_CS_SELECTIVITY
select address, hash_value, sql_id, child_number, predicate, range_id,

VIEW_NAME

VIEW_DEFINITION
---
low, high from GV$SQL_CS_SELECTIVITY where inst_id=USERENV('Instance')

V$SQL_CS_STATISTICS
select address, hash_value, sql_id, child_number,
bind_set_hash_value, peeked, executions, rows_processed,
buffer_gets, cpu_time from GV$SQL_CS_STATISTICS where
inst_id=USERENV('Instance')

V$SQL_CURSOR

VIEW_NAME

VIEW_DEFINITION
---
select CURNO , FLAG , STATUS , PARENT_HANDLE , PARENT_LOCK , CHILD_LOCK ,
CHILD_PIN , PERS_HEAP_MEM , WORK_HEAP_MEM , BIND_VARS , DEFINE_VARS ,
BIND_MEM_LOC , INST_FLAG , INST_FLAG2, CHILD_HANDLE from GV$SQL_CURSOR where
inst_id = USERENV('Instance')

V$SQL_FEATURE
select SQL_FEATURE, DESCRIPTION, PROPERTY from GV$SQL_FEATURE
where inst_id=USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$SQL_FEATURE_DEPENDENCY
select SQL_FEATURE, DEPEND_ON from GV$SQL_FEATURE_DEPENDENCY where
inst_id=USERENV('Instance')

V$SQL_FEATURE_HIERARCHY
select SQL_FEATURE, PARENT_ID from GV$SQL_FEATURE_HIERARCHY where
inst_id=USERENV('Instance')

V$SQL_HINT

VIEW_NAME

VIEW_DEFINITION

```
--------------------------------------------------------------------------------
select  NAME,       SQL_FEATURE,       CLASS,       INVERSE,
TARGET_LEVEL,       PROPERTY,       VERSION,       VERSION_OUTLINE from
gv$sql_hint where inst_id=USERENV('Instance')

V$SQL_JOIN_FILTER
SELECT qc_session_id, qc_instance_id, sql_plan_hash_value,
length, bits_set, filtered, probed, active         FROM GV$SQL_JOIN_FILTER
WHERE inst_id = USERENV('INSTANCE')


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
V$SQL_MONITOR
select  KEY,       STATUS,       FIRST_REFRESH_TIME,
LAST_REFRESH_TIME,       REFRESH_COUNT,       SID,       PROCESS_NAME,
SQL_ID,       SQL_EXEC_START,       SQL_EXEC_ID,
SQL_PLAN_HASH_VALUE,       SQL_CHILD_ADDRESS,       SESSION_SERIAL#,
PX_SERVER#,       PX_SERVER_GROUP,       PX_SERVER_SET,
PX_QCINST_ID,       PX_QCSID,       ELAPSED_TIME,       CPU_TIME,
FETCHES,       BUFFER_GETS,       DISK_READS,       DIRECT_WRITES,
APPLICATION_WAIT_TIME,       CONCURRENCY_WAIT_TIME,       CLUSTER_WAIT_TIME,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
USER_IO_WAIT_TIME,       PLSQL_EXEC_TIME,       JAVA_EXEC_TIME from
GV$SQL_MONITOR where inst_id=USERENV('Instance')

V$SQL_OPTIMIZER_ENV
select ADDRESS,                HASH_VALUE,                SQL_ID,
CHILD_ADDRESS,             CHILD_NUMBER,             ID,
NAME,                    ISDEFAULT,                VALUE
from  GV$SQL_OPTIMIZER_ENV    where  INST_ID = USERENV('Instance')


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
V$SQL_PLAN
select ADDRESS, HASH_VALUE, SQL_ID, PLAN_HASH_VALUE, CHILD_ADDRESS,
CHILD_NUMBER,  TIMESTAMP, OPERATION,
OPTIONS, OBJECT_NODE, OBJECT#, OBJECT_OWNER, OBJECT_NAME,
OBJECT_ALIAS, OBJECT_TYPE, OPTIMIZER,                                ID,
PARENT_ID, DEPTH, POSITION, SEARCH_COLUMNS, COST, CARDINALITY,         BYTES,
OTHER_TAG, PARTITION_START, PARTITION_STOP, PARTITION_ID,           OTHER,
DISTRIBUTION, CPU_COST, IO_COST, TEMP_SPACE,
ACCESS_PREDICATES, FILTER_PREDICATES, PROJECTION, TIME, QBLOCK_NAME,
```

```
VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
REMARKS, OTHER_XML
from GV$SQL_PLAN
where inst_id = USERENV('Instance')

V$SQL_PLAN_MONITOR
select  KEY,        STATUS,        FIRST_REFRESH_TIME,
LAST_REFRESH_TIME,        FIRST_CHANGE_TIME,        LAST_CHANGE_TIME,
REFRESH_COUNT,        SID,        PROCESS_NAME,        SQL_ID,
SQL_EXEC_START,        SQL_EXEC_ID,        SQL_PLAN_HASH_VALUE,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
SQL_CHILD_ADDRESS,        PLAN_LINE_ID,        PLAN_OPERATION,
PLAN_OPTIONS,        STARTS,        OUTPUT_ROWS,        WORKAREA_MEM,
WORKAREA_MAX_MEM,        WORKAREA_TEMPSEG,        WORKAREA_MAX_TEMPSEG from
GV$SQL_PLAN_MONITOR where inst_id=USERENV('Instance')

V$SQL_PLAN_STATISTICS
select ADDRESS, HASH_VALUE, SQL_ID, PLAN_HASH_VALUE, CHILD_ADDRESS,
CHILD_NUMBER, OPERATION_ID, EXECUTIONS,
LAST_STARTS, STARTS, LAST_OUTPUT_ROWS, OUTPUT_ROWS,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
LAST_CR_BUFFER_GETS, CR_BUFFER_GETS, LAST_CU_BUFFER_GETS,
CU_BUFFER_GETS, LAST_DISK_READS, DISK_READS,
LAST_DISK_WRITES, DISK_WRITES,
LAST_ELAPSED_TIME, ELAPSED_TIME                                 from
GV$SQL_PLAN_STATISTICS                                         where
inst_id = USERENV('Instance')

V$SQL_PLAN_STATISTICS_ALL
select ADDRESS, HASH_VALUE, SQL_ID, PLAN_HASH_VALUE, CHILD_ADDRESS,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
CHILD_NUMBER, TIMESTAMP, OPERATION, OPTIONS, OBJECT_NODE,
OBJECT#, OBJECT_OWNER, OBJECT_NAME, OBJECT_ALIAS,
OBJECT_TYPE, OPTIMIZER,
ID, PARENT_ID, DEPTH, POSITION, SEARCH_COLUMNS, COST,
CARDINALITY, BYTES, OTHER_TAG, PARTITION_START, PARTITION_STOP,
```

PARTITION_ID, OTHER, DISTRIBUTION, CPU_COST, IO_COST,
TEMP_SPACE, ACCESS_PREDICATES, FILTER_PREDICATES, PROJECTION,
TIME, QBLOCK_NAME, REMARKS, OTHER_XML, EXECUTIONS,
LAST_STARTS,

VIEW_NAME

VIEW_DEFINITION

STARTS, LAST_OUTPUT_ROWS, OUTPUT_ROWS, LAST_CR_BUFFER_GETS,
CR_BUFFER_GETS, LAST_CU_BUFFER_GETS, CU_BUFFER_GETS,
LAST_DISK_READS, DISK_READS, LAST_DISK_WRITES, DISK_WRITES,
LAST_ELAPSED_TIME, ELAPSED_TIME, POLICY,
ESTIMATED_OPTIMAL_SIZE, ESTIMATED_ONEPASS_SIZE,
LAST_MEMORY_USED, LAST_EXECUTION, LAST_DEGREE,
TOTAL_EXECUTIONS, OPTIMAL_EXECUTIONS, ONEPASS_EXECUTIONS,
MULTIPASSES_EXECUTIONS, ACTIVE_TIME, MAX_TEMPSEG_SIZE,
LAST_TEMPSEG_SIZE   from

VIEW_NAME

VIEW_DEFINITION

GV$SQL_PLAN_STATISTICS_ALL   where
inst_id = USERENV('Instance')

V$SQL_REDIRECTION
select ADDRESS,PARENT_HANDLE,HASH_VALUE,SQL_ID,CHILD_NUMBER, PARSING_USER_ID,
PARSING_SCHEMA_ID,COMMAND_TYPE, REASON, ERROR_CODE, POSITION, SQL_TEXT_PIECE,
ERROR_MESSAGE from GV$SQL_REDIRECTION where inst_id = USERENV('Instance')

V$SQL_SHARED_CURSOR

VIEW_NAME

VIEW_DEFINITION
----
select SQL_ID, ADDRESS, CHILD_ADDRESS, CHILD_NUMBER, UNBOUND_CURSOR,
SQL_TYPE_MISMATCH, OPTIMIZER_MISMATCH,
OUTLINE_MISMATCH, STATS_ROW_MISMATCH, LITERAL_MISMATCH, FORCE_HARD_PARSE,
EXPLAIN_PLAN_CURSOR, BUFFERED_DML_MISMATCH, PDML_ENV_MISMATCH,
INST_DRTLD_MISMATCH, SLAVE_QC_MISMATCH, TYPECHECK_MISMATCH,
AUTH_CHECK_MISMATCH, BIND_MISMATCH, DESCRIBE_MISMATCH, LANGUAGE_MISMATCH,
TRANSLATION_MISMATCH, ROW_LEVEL_SEC_MISMATCH, INSUFF_PRIVS,
INSUFF_PRIVS_REM, REMOTE_TRANS_MISMATCH, LOGMINER_SESSION_MISMATCH,
INCOMP_LTRL_MISMATCH, OVERLAP_TIME_MISMATCH, EDITION_MISMATCH,

VIEW_NAME

VIEW_DEFINITION

MV_QUERY_GEN_MISMATCH, USER_BIND_PEEK_MISMATCH, TYPCHK_DEP_MISMATCH,

NO_TRIGGER_MISMATCH, FLASHBACK_CURSOR, ANYDATA_TRANSFORMATION,
INCOMPLETE_CURSOR, TOP_LEVEL_RPI_CURSOR, DIFFERENT_LONG_LENGTH,
LOGICAL_STANDBY_APPLY, DIFF_CALL_DURN, BIND_UACS_DIFF,
PLSQL_CMP_SWITCHS_DIFF, CURSOR_PARTS_MISMATCH, STB_OBJECT_MISMATCH,
CROSSEDITION_TRIGGER_MISMATCH, PQ_SLAVE_MISMATCH, TOP_LEVEL_DDL_MISMATCH,
MULTI_PX_MISMATCH, BIND_PEEKED_PQ_MISMATCH, MV_REWRITE_MISMATCH,
ROLL_INVALID_MISMATCH, OPTIMIZER_MODE_MISMATCH,
PX_MISMATCH, MV_STALEOBJ_MISMATCH, FLASHBACK_TABLE_MISMATCH,

VIEW_NAME

VIEW_DEFINITION

LITREP_COMP_MISMATCH, PLSQL_DEBUG, LOAD_OPTIMIZER_STATS, ACL_MISMATCH,
FLASHBACK_ARCHIVE_MISMATCH, LOCK_USER_SCHEMA_FAILED,
REMOTE_MAPPING_MISMATCH, LOAD_RUNTIME_HEAP_FAILED
from GV$SQL_SHARED_CURSOR
where inst_id = USERENV('Instance')

V$SQL_SHARED_MEMORY
select SQL_TEXT, SQL_FULLTEXT, HASH_VALUE, SQL_ID, HEAP_DESC, STRUCTURE,
FUNCTION, CHUNK_COM , CHUNK_PTR , CHUNK_SIZE , ALLOC_CLASS , CHUNK_TYPE ,

VIEW_NAME

VIEW_DEFINITION

SUBHEAP_DESC from GV$SQL_SHARED_MEMORY where inst_id = USERENV('Instance')

V$SQL_WORKAREA
select ADDRESS, HASH_VALUE, SQL_ID, CHILD_NUMBER,
WORKAREA_ADDRESS,
OPERATION_TYPE, OPERATION_ID, POLICY, ESTIMATED_OPTIMAL_SIZE,
ESTIMATED_ONEPASS_SIZE, LAST_MEMORY_USED, LAST_EXECUTION,
LAST_DEGREE, TOTAL_EXECUTIONS, OPTIMAL_EXECUTIONS,
ONEPASS_EXECUTIONS, MULTIPASSES_EXECUTIONS, ACTIVE_TIME,

VIEW_NAME

VIEW_DEFINITION

MAX_TEMPSEG_SIZE, LAST_TEMPSEG_SIZE   from
GV$SQL_WORKAREA   where
inst_id = USERENV('Instance')

V$SQL_WORKAREA_ACTIVE
select SQL_HASH_VALUE,   SQL_ID,   SQL_EXEC_START,
SQL_EXEC_ID,   WORKAREA_ADDRESS,   OPERATION_TYPE,
OPERATION_ID,   POLICY,   SID,   QCINST_ID,
QCSID,   ACTIVE_TIME,   WORK_AREA_SIZE,

VIEW_NAME

VIEW_DEFINITION

EXPECTED_SIZE, ACTUAL_MEM_USED, MAX_MEM_USED,
NUMBER_PASSES, TEMPSEG_SIZE, TABLESPACE,
SEGRFNO#, SEGBLK# from GV$SQL_WORKAREA_ACTIVE where INST_ID =
USERENV('Instance')

V$SQL_WORKAREA_HISTOGRAM
select LOW_OPTIMAL_SIZE, HIGH_OPTIMAL_SIZE,
OPTIMAL_EXECUTIONS, ONEPASS_EXECUTIONS,
MULTIPASSES_EXECUTIONS, TOTAL_EXECUTIONS from

VIEW_NAME

VIEW_DEFINITION

GV$SQL_WORKAREA_HISTOGRAM where INST_ID = USERENV('Instance')

V$SSCR_SESSIONS
SELECT sid, serial#, state, crmode, scope, nc_component, nc_reason, options,
timeout FROM gv$sscr_sessions WHERE inst_id = USERENV('Instance')

V$STANDBY_APPLY_SNAPSHOT
select SNAPSHOT_TIME, SESSION_ID, THREAD#, RESET_TIMESTAMP, SEQUENCE#, BLOCK#,
APPLIED_SCN, APPLIED_TIME, NEWEST_RESET_TIMESTAMP, NEWEST_ARCHIVED_SEQ#,

VIEW_NAME

VIEW_DEFINITION

NEWEST_TIME, NEWEST_USED, NEWEST_SRL_SEQ#, BLOCKSIZE, APPLY_RATE FROM
GV$STANDBY_APPLY_SNAPSHOT where INST_ID = USERENV('INSTANCE')

V$STANDBY_LOG
select GROUP# , DBID , THREAD# , SEQUENCE# , BYTES , USED , ARCHIVED , STATUS ,
FIRST_CHANGE# , FIRST_TIME , LAST_CHANGE# , LAST_TIME from GV$STANDBY_LOG
where inst_id = USERENV('Instance')

V$STATISTICS_LEVEL

VIEW_NAME

VIEW_DEFINITION
---
select statistics_name, description, session_status, system_status,
activation_level, statistics_view_name, session_settable from
gv$statistics_level where inst_id = USERENV('Instance')

V$STATNAME
select STATISTIC# , NAME , CLASS, STAT_ID from GV$STATNAME where inst_id =
USERENV('Instance')

V$STREAMS_APPLY_COORDINATOR

VIEW_NAME

VIEW_DEFINITION

SELECT SID, SERIAL#, STATE, APPLY#, APPLY_NAME,
TOTAL_APPLIED,TOTAL_WAIT_DEPS,TOTAL_WAIT_COMMITS, TOTAL_ADMIN, TOTAL_ASSIGNED,
TOTAL_RECEIVED,TOTAL_IGNORED, TOTAL_ROLLBACKS, TOTAL_ERRORS,LWM_TIME,
LWM_MESSAGE_NUMBER, LWM_MESSAGE_CREATE_TIME,HWM_TIME, HWM_MESSAGE_NUMBER,

HWM_MESSAGE_CREATE_TIME,STARTUP_TIME, ELAPSED_SCHEDULE_TIME, ELAPSED_IDLE_TIME

from GV$STREAMS_APPLY_COORDINATOR WHERE INST_ID = USERENV('Instance')

V$STREAMS_APPLY_READER
select SID, SERIAL#, APPLY#, APPLY_NAME, STATE, TOTAL_MESSAGES_DEQUEUED,

VIEW_NAME

VIEW_DEFINITION

TOTAL_MESSAGES_SPILLED, DEQUEUE_TIME, DEQUEUED_MESSAGE_NUMBER,
DEQUEUED_MESSAGE_CREATE_TIME, SGA_USED, ELAPSED_DEQUEUE_TIME,
ELAPSED_SCHEDULE_TIME, ELAPSED_SPILL_TIME, LAST_BROWSE_NUM, OLDEST_SCN_NUM,
LAST_BROWSE_SEQ, LAST_DEQ_SEQ, OLDEST_XIDUSN, OLDEST_XIDSLT, OLDEST_XIDSQN,
SPILL_LWM_SCN, PROXY_SID, PROXY_SERIAL, PROXY_SPID, CAPTURE_BYTES_RECEIVED from
GV$STREAMS_APPLY_READER where INST_ID = USERENV('Instance')

V$STREAMS_APPLY_SERVER
select SID, SERIAL#, APPLY#, APPLY_NAME,SERVER_ID, STATE, XIDUSN, XIDSLT,

VIEW_NAME

VIEW_DEFINITION

XIDSQN, COMMITSCN,DEP_XIDUSN, DEP_XIDSLT, DEP_XIDSQN, DEP_COMMITSCN,
MESSAGE_SEQUENCE,TOTAL_ASSIGNED, TOTAL_ADMIN,
TOTAL_ROLLBACKS,TOTAL_MESSAGES_APPLIED, APPLY_TIME, APPLIED_MESSAGE_NUMBER,
APPLIED_MESSAGE_CREATE_TIME,ELAPSED_DEQUEUE_TIME, ELAPSED_APPLY_TIME from
GV$STREAMS_APPLY_SERVER where INST_ID = USERENV('Instance')

V$STREAMS_CAPTURE
select SID, SERIAL#, CAPTURE#, CAPTURE_NAME, LOGMINER_ID,   STARTUP_TIME,
STATE, TOTAL_PREFILTER_DISCARDED, TOTAL_PREFILTER_KEPT,

VIEW_NAME

VIEW_DEFINITION

TOTAL_PREFILTER_EVALUATIONS,TOTAL_MESSAGES_CAPTURED, CAPTURE_TIME,

CAPTURE_MESSAGE_NUMBER, CAPTURE_MESSAGE_CREATE_TIME,
TOTAL_MESSAGES_CREATED, TOTAL_FULL_EVALUATIONS, TOTAL_MESSAGES_ENQUEUED,
ENQUEUE_TIME, ENQUEUE_MESSAGE_NUMBER, ENQUEUE_MESSAGE_CREATE_TIME,
AVAILABLE_MESSAGE_NUMBER, AVAILABLE_MESSAGE_CREATE_TIME,
ELAPSED_CAPTURE_TIME, ELAPSED_RULE_TIME,ELAPSED_ENQUEUE_TIME,
ELAPSED_LCR_TIME, ELAPSED_REDO_WAIT_TIME, ELAPSED_PAUSE_TIME,
STATE_CHANGED_TIME, APPLY_NAME, APPLY_DBLINK, APPLY_MESSAGES_SENT,
APPLY_BYTES_SENT from GV$STREAMS_CAPTURE where INST_ID = USERENV('Instance')

VIEW_NAME

VIEW_DEFINITION

V$STREAMS_MESSAGE_TRACKING
select TRACKING_LABEL, TAG, COMPONENT_NAME, COMPONENT_TYPE, ACTION,
ACTION_DETAILS, TIMESTAMP, MESSAGE_CREATION_TIME,
MESSAGE_NUMBER, TRACKING_ID, SOURCE_DATABASE_NAME, OBJECT_OWNER,
OBJECT_NAME, XID, COMMAND_TYPE from
GV$STREAMS_MESSAGE_TRACKING where INST_ID = USERENV('Instance')

V$STREAMS_POOL_ADVICE

VIEW_NAME

VIEW_DEFINITION
---
select streams_pool_size_for_estimate, streams_pool_size_factor,
estd_spill_count, estd_spill_time, estd_unspill_count, estd_unspill_time from
gv$streams_pool_advice where inst_id = USERENV('Instance')

V$STREAMS_TRANSACTION
select STREAMS_NAME, STREAMS_TYPE, XIDUSN, XIDSLT, XIDSQN,
CUMULATIVE_MESSAGE_COUNT, TOTAL_MESSAGE_COUNT,
FIRST_MESSAGE_TIME, FIRST_MESSAGE_NUMBER, LAST_MESSAGE_TIME,
LAST_MESSAGE_NUMBER from GV$STREAMS_TRANSACTION where

VIEW_NAME

VIEW_DEFINITION

INST_ID = USERENV('Instance')

V$SUBCACHE
select OWNER_NAME , NAME , TYPE , HEAP_NUM , CACHE_ID , CACHE_CNT , HEAP_SZ ,
HEAP_ALOC , HEAP_USED from GV$SUBCACHE where inst_id = USERENV('Instance')

V$SUBSCR_REGISTRATION_STATS
select reg_id, num_ntfns, num_grouping_ntfns, last_ntfn_start_time,
last_ntfn_sent_time, total_emon_latency, emon#, all_emon_servers,

VIEW_NAME

```
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
total_payload_bytes_sent, num_retries, total_plsql_exec_time,        last_err,
last_err_time, last_update_time  from gv$subscr_registration_stats where inst_id
= USERENV('Instance')

V$SYSAUX_OCCUPANTS
SELECT occupant_name, occupant_desc, schema_name,
move_procedure, move_procedure_desc, space_usage_kbytes         FROM
gv$sysaux_occupants         WHERE  inst_id = USERENV('INSTANCE')


VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
V$SYSMETRIC
SELECT begin_time, end_time, intsize_csec,          group_id, metric_id,
metric_name,          value, metric_unit        FROM   gv$sysmetric
WHERE  inst_id = USERENV('INSTANCE')

V$SYSMETRIC_HISTORY
SELECT begin_time, end_time, intsize_csec,          group_id, metric_id,
metric_name,          value, metric_unit        FROM   gv$sysmetric_history
WHERE  inst_id = USERENV('INSTANCE')

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------

V$SYSMETRIC_SUMMARY
SELECT begin_time, end_time, intsize_csec,          group_id, metric_id,
metric_name, num_interval, maxval, minval,          average,
standard_deviation, metric_unit         FROM   gv$sysmetric_summary
WHERE  inst_id = USERENV('INSTANCE')

V$SYSSTAT
select  STATISTIC# , NAME , CLASS , VALUE, STAT_ID from GV$SYSSTAT where inst_id

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
= USERENV('Instance')

V$SYSTEM_CURSOR_CACHE
select  OPENS , HITS , HIT_RATIO from GV$SYSTEM_CURSOR_CACHE where inst_id =
USERENV('Instance')

V$SYSTEM_EVENT
```

select event,total_waits,total_timeouts,time_waited,average_wait,
time_waited_micro, total_waits_fg, total_timeouts_fg, time_waited_fg,

VIEW_NAME

VIEW_DEFINITION
---
average_wait_fg, time_waited_micro_fg, event_id, wait_class_id, wait_class#,
wait_class from gv$system_event where inst_id = USERENV('Instance')

V$SYSTEM_FIX_CONTROL
select BUGNO, VALUE, SQL_FEATURE, DESCRIPTION,
OPTIMIZER_FEATURE_ENABLE, EVENT, IS_DEFAULT from
GV$SYSTEM_FIX_CONTROL where inst_id=USERENV('Instance')

V$SYSTEM_PARAMETER

VIEW_NAME

VIEW_DEFINITION
---
select NUM , NAME , TYPE , VALUE , DISPLAY_VALUE, ISDEFAULT , ISSES_MODIFIABLE
, ISSYS_MODIFIABLE , ISINSTANCE_MODIFIABLE, ISMODIFIED , ISADJUSTED ,
ISDEPRECATED, ISBASIC, DESCRIPTION, UPDATE_COMMENT, HASH from
GV$SYSTEM_PARAMETER where inst_id = USERENV('Instance')

V$SYSTEM_PARAMETER2
select NUM, NAME, TYPE, VALUE, DISPLAY_VALUE, ISDEFAULT, ISSES_MODIFIABLE,
ISSYS_MODIFIABLE , ISINSTANCE_MODIFIABLE, ISMODIFIED , ISADJUSTED ,
ISDEPRECATED, ISBASIC, DESCRIPTION, ORDINAL, UPDATE_COMMENT from

VIEW_NAME

VIEW_DEFINITION

GV$SYSTEM_PARAMETER2 where inst_id = USERENV('Instance')

V$SYSTEM_PARAMETER4
select SID, NUM, NAME, TYPE, DISPLAY_VALUE, ORDINAL, UPDATE_COMMENT from
GV$SYSTEM_PARAMETER4 where INST_id = USERENV('Instance')

V$SYSTEM_WAIT_CLASS
select wait_class_id, wait_class#,wait_class,total_waits,time_waited,
total_waits_fg, time_waited_fg from gv$system_wait_class where inst_id =

VIEW_NAME

VIEW_DEFINITION

USERENV('Instance')

V$SYS_OPTIMIZER_ENV

```
select ID,              NAME,            SQL_FEATURE,
ISDEFAULT,                VALUE,                    DEFAULT_VALUE
from   GV$SYS_OPTIMIZER_ENV     where  INST_ID = USERENV('Instance')
```

V$SYS_TIME_MODEL

```
select STAT_ID, STAT_NAME, VALUE from GV$SYS_TIME_MODEL  where inst_id =
```

VIEW_NAME

VIEW_DEFINITION

```
USERENV('Instance')
```

V$TABLESPACE

```
select  TS# , NAME, INCLUDED_IN_DATABASE_BACKUP, BIGFILE, FLASHBACK_ON,
ENCRYPT_IN_BACKUP from GV$TABLESPACE where inst_id = USERENV('Instance')
```

V$TEMPFILE

```
select  FILE# , CREATION_CHANGE# , CREATION_TIME , TS# , RFILE# , STATUS ,
ENABLED , BYTES, BLOCKS, CREATE_BYTES , BLOCK_SIZE , NAME from GV$TEMPFILE where
```

VIEW_NAME

VIEW_DEFINITION

```
inst_id = USERENV('Instance')
```

V$TEMPORARY_LOBS

```
select SID, CACHE_LOBS, NOCACHE_LOBS, ABSTRACT_LOBS           from
GV$TEMPORARY_LOBS           where inst_id = USERENV('Instance')
```

V$TEMPSTAT

```
select  FILE# , PHYRDS , PHYWRTS , PHYBLKRD , PHYBLKWRT , SINGLEBLKRDS,  READTIM
, WRITETIM, SINGLEBLKRDTIM, AVGIOTIM, LSTIOTIM, MINIOTIM, MAXIORTM,  MAXIOWTM
```

VIEW_NAME

VIEW_DEFINITION

```
from GV$TEMPSTAT where inst_id = USERENV('Instance')
```

V$TEMP_CACHE_TRANSFER

```
select  file_number,                                  x_2_null,
x_2_null_forced_write, x_2_null_forced_stale,          x_2_s,
x_2_s_forced_write,                                 s_2_null,
s_2_null_forced_stale,                             rbr, rbr_forced_write,
null_2_x, s_2_x, null_2_s                                 from
gv$temp_cache_transfer                                where inst_id =
```

VIEW_NAME

VIEW_DEFINITION

```
--------------------------------------------------------------------------------
USERENV('Instance')

V$TEMP_EXTENT_MAP
select  TABLESPACE_NAME , FILE_ID , BLOCK_ID , BYTES , BLOCKS , OWNER ,
RELATIVE_FNO from GV$TEMP_EXTENT_MAP where inst_id = USERENV('Instance')

V$TEMP_EXTENT_POOL
select  TABLESPACE_NAME , FILE_ID , EXTENTS_CACHED , EXTENTS_USED ,
BLOCKS_CACHED , BLOCKS_USED , BYTES_CACHED , BYTES_USED , RELATIVE_FNO from

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
GV$TEMP_EXTENT_POOL where inst_id = USERENV('Instance')

V$TEMP_PING
select  file_number, frequency,                          x_2_null,
x_2_null_forced_write, x_2_null_forced_stale,          x_2_s,
x_2_s_forced_write, x_2_ssx, x_2_ssx_forced_write,        s_2_null,
s_2_null_forced_stale, ss_2_null, ss_2_rls, wrb, wrb_forced_write, rbr,
rbr_forced_write, rbr_forced_stale, cbr, cbr_forced_write,    null_2_x, s_2_x,
ssx_2_x, null_2_s, null_2_ss, op_2_ss           from gv$temp_ping

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
where inst_id = USERENV('Instance')

V$TEMP_SPACE_HEADER
select  TABLESPACE_NAME , FILE_ID , BYTES_USED , BLOCKS_USED , BYTES_FREE ,
BLOCKS_FREE , RELATIVE_FNO from GV$TEMP_SPACE_HEADER where inst_id =
USERENV('Instance')

V$THREAD
select  THREAD# , STATUS , ENABLED , GROUPS , INSTANCE , OPEN_TIME ,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
CURRENT_GROUP# , SEQUENCE# , CHECKPOINT_CHANGE# , CHECKPOINT_TIME ,
ENABLE_CHANGE# , ENABLE_TIME , DISABLE_CHANGE# , DISABLE_TIME,
LAST_REDO_SEQUENCE#, LAST_REDO_BLOCK, LAST_REDO_CHANGE#, LAST_REDO_TIME from
GV$THREAD where inst_id = USERENV('Instance')

V$THRESHOLD_TYPES
SELECT metrics_id, metrics_group_id, operator_mask, object_type,
alert_reason_id, metric_value_type         FROM gv$threshold_types
WHERE inst_id = USERENV('INSTANCE')
```

VIEW_NAME

VIEW_DEFINITION

V$TIMER
select HSECS from GV$TIMER where inst_id = USERENV('Instance')

V$TIMEZONE_FILE
select FILENAME, VERSION from GV$TIMEZONE_FILE

V$TIMEZONE_NAMES
select TZNAME, TZABBREV from GV$TIMEZONE_NAMES

VIEW_NAME

VIEW_DEFINITION

V$TRANSACTION
select ADDR , XIDUSN , XIDSLOT , XIDSQN , UBAFIL , UBABLK , UBASQN , UBAREC ,
STATUS , START_TIME , START_SCNB , START_SCNW , START_UEXT , START_UBAFIL ,
START_UBABLK , START_UBASQN , START_UBAREC , SES_ADDR , FLAG , SPACE , RECURSIVE
, NOUNDO , PTX , NAME,PRV_XIDUSN , PRV_XIDSLT , PRV_XIDSQN , PTX_XIDUSN ,
PTX_XIDSLT , PTX_XIDSQN , "DSCN-B" , "DSCN-W" , USED_UBLK , USED_UREC , LOG_IO ,
PHY_IO , CR_GET , CR_CHANGE, START_DATE, DSCN_BASE, DSCN_WRAP, START_SCN,
DEPENDENT_SCN, XID, PRV_XID, PTX_XID from gv$transaction where inst_id =

VIEW_NAME

VIEW_DEFINITION

USERENV('Instance')

V$TRANSACTION_ENQUEUE
select ADDR , KADDR , SID , TYPE , ID1 , ID2 , LMODE , REQUEST , CTIME , BLOCK
from GV$TRANSACTION_ENQUEUE where inst_id = USERENV('Instance')

V$TRANSPORTABLE_PLATFORM
SELECT PLATFORM_ID, PLATFORM_NAME, decode(endian_format, 1,'Big'
,0,'Little','UNKNOWN FORMAT') FROM x$kcpxpl

VIEW_NAME

VIEW_DEFINITION

V$TSM_SESSIONS
select sid, serial#, state, migratable, migration_allowed, transferable,
migration_boundary, nonmigratability_reason, nonmigratability_info,
nontransferability_reason, nontransferability_info, stateless, preserve_state,

cost, destination, roundtrips, blocking_component, start_time, sequence_number
from gv$tsm_sessions where inst_id = USERENV('Instance')

V$TYPE_SIZE

VIEW_NAME

VIEW_DEFINITION
----
select COMPONENT , TYPE , DESCRIPTION , TYPE_SIZE from GV$TYPE_SIZE where
inst_id = USERENV('Instance')

V$UNDOSTAT
select   to_date(KTUSMSTRBEGTIME,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),   to_date(KTUSMSTRENDTIME,'MM/DD/RR
HH24:MI:SS','NLS_CALENDAR=Gregorian'),   KTUSMSTTSN, KTUSMSTUSU, KTUSMSTTCT,
KTUSMSTMQL, KTUSMSTRMQI, KTUSMSTMTC,   KTUSMSTUAC, KTUSMSTUBS, KTUSMSTUBR,
KTUSMSTXAC, KTUSMSTXBS, KTUSMSTXBR,   KTUSMSTSOC, KTUSMSTOOS, KTUSMSTABK,

VIEW_NAME

VIEW_DEFINITION

KTUSMSTUBK, KTUSMSTEBK, KTUSMSTTUR   from X$KTUSMST where INST_ID =
userenv('instance')

V$UNUSABLE_BACKUPFILE_DETAILS
select b.session_recid session_key, b.session_recid, b.session_stamp,   a.* from
(select a.rman_status_recid, a.rman_status_stamp,   'BACKUPSET' btype,
b.recid btype_key, a.set_stamp id1, a.set_count id2,   'BACKUPPIECE'
filetype,b.recid filetype_key,a.status,a.bytes filesize,   a.device_type,
a.handle filename, a.media, a.media_pool   from v$backup_piece a,

VIEW_NAME

VIEW_DEFINITION
----
v$backup_set b where   a.set_stamp = b.set_stamp and a.set_count =
b.set_count and   status <> 'A'   union   select rman_status_recid,
rman_status_stamp,   'IMAGECOPY', recid, recid, stamp, 'DATAFILECOPY',
recid, status,   (blocks+1)*block_size, 'DISK', name, null, null
from v$datafile_copy where status <> 'A' and file#<>0   union   select
rman_status_recid, rman_status_stamp,   'IMAGECOPY', recid, recid, stamp,
'CONTROLFILECOPY', recid, status,   (blocks+1)*block_size, 'DISK', name,
null, null   from v$datafile_copy where status <> 'A' and file#=0   union
select rman_status_recid, rman_status_stamp,   'PROXYCOPY', recid, recid,

VIEW_NAME

VIEW_DEFINITION
----
stamp, 'DATAFILECOPY', recid, status,   (blocks+1)*block_size,

device_type, handle, media, media_pool from v$proxy_datafile where status
<> 'A' and file#<>0 union select rman_status_recid, rman_status_stamp,
'PROXYCOPY', recid, recid, stamp, 'CONTROLFILECOPY', recid, status,
(blocks+1)*block_size, device_type, handle, media, media_pool from
v$proxy_datafile where status <> 'A' and file#=0 union select
rman_status_recid, rman_status_stamp, 'PROXYCOPY', recid, recid, stamp,
'ARCHIVELOGCOPY', recid, status, (blocks+1)*block_size, device_type,
handle, media, media_pool from v$proxy_archivedlog where status <> 'A') a,

VIEW_NAME

VIEW_DEFINITION
---
v$rman_status b, (select /*+ no_merge */ sys.dbms_rcvman.sv_getsessionkey skey
from dual) c, (select /*+ no_merge */
sys.dbms_rcvman.sv_getsessionfromTimeRange fTime from dual) d, (select
/*+ no_merge */ sys.dbms_rcvman.sv_getsessionuntilTimeRange uTime from
dual) e where a.rman_status_recid = b.recid (+) and
a.rman_status_stamp = b.stamp (+) and (c.skey is null or c.skey =
b.session_recid) and (d.fTime is null or d.fTime <= b.start_time) and
(e.uTime is null or e.uTime >= b.end_time)

VIEW_NAME

VIEW_DEFINITION

V$VERSION
select BANNER from GV$VERSION where inst_id = USERENV('Instance')

V$VPD_POLICY
select ADDRESS,PARADDR,SQL_HASH,SQL_ID,CHILD_NUMBER, OBJECT_OWNER,
OBJECT_NAME,POLICY_GROUP,POLICY,POLICY_FUNCTION_OWNER,PREDICATE from
GV$VPD_POLICY where inst_id = USERENV('Instance')

V$WAITCLASSMETRIC

VIEW_NAME

VIEW_DEFINITION

SELECT begin_time, end_time, intsize_csec, wait_class#,
wait_class_id, average_waiter_count, dbtime_in_wait, time_waited,
wait_count FROM gv$waitclassmetric WHERE inst_id =
USERENV('INSTANCE')

V$WAITCLASSMETRIC_HISTORY
SELECT begin_time, end_time, intsize_csec, wait_class#,
wait_class_id, average_waiter_count, dbtime_in_wait, time_waited,
wait_count FROM gv$waitclassmetric_history WHERE inst_id =

VIEW_NAME

```
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
USERENV('INSTANCE')

V$WAITSTAT
select class,count,time from gv$waitstat where inst_id = USERENV('Instance')

V$WAIT_CHAINS
select  s.chain_id,  decode(s.chain_is_cycle, 0,'FALSE','TRUE'),
s.chain_signature, s.chain_signature_hash,   s.instance, s.osid, s.pid, s.sid,
s.sess_serial#,   decode(s.blocker_is_valid, 0,'FALSE','TRUE'),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
decode(s.blocker_is_valid, 0, to_number(null), s.blocker_instance),
s.blocker_osid,  decode(s.blocker_is_valid, 0, to_number(null), s.blocker_pid),
decode(s.blocker_is_valid, 0, to_number(null), s.blocker_sid),
decode(s.blocker_is_valid, 0, to_number(null), s.blocker_sess_serial#),
decode(s.blocker_chain_id, 0, to_number(null), s.blocker_chain_id),
decode(s.in_wait, 0,'FALSE','TRUE'),   decode(s.in_wait, 0,
s.time_since_last_wait_secs, to_number(null)),   decode(s.in_wait, 0,
to_number(null), s.wait_id),  decode(s.in_wait, 0, to_number(null),
s.wait_event),  s.wait_event_text,  decode(s.in_wait, 0, to_number(null), s.p1),

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
s.p1_text,  decode(s.in_wait, 0, to_number(null), s.p2),  s.p2_text,
decode(s.in_wait, 0, to_number(null), s.p3),  s.p3_text,   decode(s.in_wait, 0,
to_number(null), s.in_wait_secs),  decode(s.in_wait, 0, to_number(null),
s.time_remaining_secs),  s.num_waiters,   decode(s.in_wait, 0, to_number(null),
s.row_wait_obj#),  decode(s.in_wait, 0, to_number(null), s.row_wait_file#),
decode(s.in_wait, 0, to_number(null), s.row_wait_block#),  decode(s.in_wait, 0,
to_number(null), s.row_wait_row#)  from X$KSDHNG_CHAINS s

V$WALLET

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
SELECT        CERT_ID, DN, SERIAL_NUM, ISSUER, KEYSIZE, STATUS        FROM
GV$WALLET        WHERE inst_id = USERENV('INSTANCE')

V$WORKLOAD_REPLAY_THREAD
SELECT clock, next_ticker,       sid, serial#, spid, logon_user, logon_time,
event, event_id, event#,       p1text, p1, p2text, p2, p3text, p3,
wait_for_scn,       file_id, call_counter,       dependent_scn, statement_scn,
```

```
commit_wait_scn, post_commit_scn,       action_type, session_type, wrc_id,
file_name,      skip_it, dirty_buffers,       dbtime, network_time,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
think_time,       time_gain, time_loss, user_calls,       client_os_user,
client_host, client_pid, program FROM   GV$WORKLOAD_REPLAY_THREAD WHERE  inst_id
= USERENV('INSTANCE')

V$XML_AUDIT_TRAIL
select AUDIT_TYPE, SESSION_ID, PROXY_SESSIONID, STATEMENTID, ENTRYID,
EXTENDED_TIMESTAMP, GLOBAL_UID, DB_USER, CLIENTIDENTIFIER, EXT_NAME,
OS_USER, OS_HOST, OS_PROCESS, TERMINAL, INSTANCE_NUMBER,       OBJECT_SCHEMA,
OBJECT_NAME, POLICY_NAME, NEW_OWNER, NEW_NAME, ACTION,       STATEMENT_TYPE,

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
TRANSACTIONID, RETURNCODE, SCN, COMMENT_TEXT,       AUTH_PRIVILEGES, GRANTEE,
PRIV_USED, SES_ACTIONS, OS_PRIVILEGE,       ECONTEXT_ID, SQL_BIND, SQL_TEXT,
OBJ_EDITION_NAME from GV$XML_AUDIT_TRAIL where inst_id=USERENV('Instance')

V$_LOCK
select  LADDR , KADDR , SADDR , RADDR , LMODE , REQUEST , CTIME , BLOCK from
GV$_LOCK where inst_id = USERENV('Instance')

V$_LOCK1

VIEW_NAME
------------------------------
VIEW_DEFINITION
--------------------------------------------------------------------------------
select  LADDR , KADDR , SADDR , RADDR , LMODE , REQUEST , CTIME , BLOCK from
GV$_LOCK1 where inst_id = USERENV('Instance')

V$_SEQUENCES
select  SEQUENCE_OWNER , SEQUENCE_NAME , OBJECT# , ACTIVE_FLAG , REPLENISH_FLAG
, WRAP_FLAG , NEXTVALUE , MIN_VALUE , MAX_VALUE , INCREMENT_BY , CYCLE_FLAG ,
ORDER_FLAG , CACHE_SIZE , HIGHWATER , BACKGROUND_INSTANCE_LOCK ,
INSTANCE_LOCK_FLAGS from GV$_SEQUENCES where inst_id = USERENV('Instance')


943 rows selected.

SQL> spool off
```