How to Query Database Information

Commands and Scripts with Examples

Asfaw Gedamu

20/02/2024

Querying Database Information

This guide outlines procedures and scripts for querying various database information using SQL commands and scripts.

Note: Replace <username> and <password> with your actual credentials and adjust commands based on your specific database environment.

Privileges Required:

Most of the following queries require **SELECT** privilege on relevant tables and views. Some queries might require additional privileges depending on the specific database and desired information.

Executing Saved Scripts:

1. **Save the script** as a .sql file.
2. **Connect** to your database using a SQL client tool (e.g., SQL*Plus, pgAdmin).
3. **Execute the script** using the appropriate command in your client tool. For example, in SQL*Plus: @your_script.sql

Note that I have provided two scripts:

Comparison:

- **Focus:** Script 1 is more specific, targeting relevant information for analyzing redo log members. Script 2 provides comprehensive details but might be overwhelming for quick analysis.
- **Readability:** Script 1 enhances readability with formatting and filtering, while Script 2 requires scrolling through potentially irrelevant data.
- **Performance:** Script 1 might be slightly slower due to filtering and joining tables, while Script 2 is faster as it retrieves all data directly.

Choosing the optimal script depends on your specific needs:

- **If you need a quick overview of all redo log members:** Use Script 2.
- **If you want to focus on specific details and improve readability:** Use Script 1.

Additional considerations:

- **Customization:** Script 1 can be further customized by adjusting the selected columns, formatting options, and filtering criteria based on your specific requirements.
- **Database environment:** Ensure compatibility of the scripts with your specific database version and configuration.

1. Get Redo Log Member Information:

Script 1:

```
col member for a56
set pagesize 299
set lines 299
select l.group#, l.thread#,
f.member,
l.archived,
l.status,
(bytes/1024/1024) "Size (MB)"
from
v$log l, v$logfile f
where f.group# = l.group#
order by 1,2
/
```

Script 2 :

```
SELECT * FROM v$logfile;
```

2. Get DDL of All Tablespaces:

Script 1:

```
set heading off;
set echo off;
Set pages 999;
set long 90000;
```

```
spool ddl_tablespace.sql
select dbms_metadata.get_ddl('TABLESPACE',tb.tablespace_name)
from dba_tablespaces tb;
spool off
```

Script 2 :

```
SELECT dbms_metadata.get_ddl('TABLESPACE', tablespace_name) AS
ddl
FROM dba_tablespaces;
```

3. Get DDL of All Privileges Granted to User:

Script 1:

```
set feedback off pages 0 long 900000 lines 20000 pagesize 20000
serveroutput on
accept USERNAME prompt "Enter username :"
--This line add a semicolon at the end of each statement
execute
dbms_METADATA.SET_TRANSFORM_PARAM(DBMS_METADATA.SESSION_TRANSFOR
M,'SQLTERMINATOR',true);
-- This will generate the DDL for the user and add his
objects,system and role grants
SELECT DBMS_METADATA.GET_DDL('USER',username) as script from
DBA_USERS where username='&username'
UNION ALL
SELECT DBMS_METADATA.GET_GRANTED_DDL('SYSTEM_GRANT',grantee)as
script from DBA_SYS_PRIVS where grantee='&username' and rownum=1
UNION ALL
SELECT DBMS_METADATA.GET_GRANTED_DDL('ROLE_GRANT',grantee)as
script from DBA_ROLE_PRIVS where grantee='&username' and
rownum=1
UNION ALL
SELECT DBMS_METADATA.GET_GRANTED_DDL('OBJECT_GRANT',grantee)as
script from DBA_TAB_PRIVS where grantee='&username' and
rownum=1;
```

Script 2:

```
SELECT * FROM user_grants;
```

4. Get Size of the Database:

Script 1:

```
col "Database Size" format a20
col "Free space" format a20
col "Used space" format a20
select round(sum(used.bytes) / 1024 / 1024 / 1024 ) || ' GB'
"Database Size"
, round(sum(used.bytes) / 1024 / 1024 / 1024 ) -
round(free.p / 1024 / 1024 / 1024) || ' GB' "Used space"
, round(free.p / 1024 / 1024 / 1024) || ' GB' "Free space"
from (select bytes
from v$datafile
union all
select bytes
from v$tempfile
union all
select bytes
from v$log) used
, (select sum(bytes) as p
from dba_free_space) free
group by free.p
/
```

Script 2:

```
SELECT SUM(bytes) / (1024 * 1024 * 1024) AS size_gb
FROM dba_data_files;
```

5. View Hidden Parameter Setting:

Script 1:

```
Set lines 2000
col NAME for a45
col DESCRIPTION for a100
SELECT name,description from SYS.V$PARAMETER WHERE name LIKE
'\_%' ESCAPE '\'
/
```

Script 2:

```
SELECT name, value FROM v$parameter WHERE name =
'<parameter_name>';
```

6. Get ACL Details in Database:

Script 1:

```
set lines 200
COL ACL_OWNER FOR A12
COL ACL FOR A67
COL HOST FOR A34
col PRINCIPAL for a20
col PRIVILEGE for a13
select ACL_OWNER,ACL,HOST,LOWER_PORT,UPPER_PORT FROM
dba_network_acls;
select ACL_OWNER,ACL,PRINCIPAL,PRIVILEGE from
dba_network_acl_privileges;
```

Script 2:

```
SELECT * FROM dba_users_roles;
```

7. Archive Generation per Hour:

Script 1:

```
set lines 299
SELECT TO_CHAR(TRUNC(FIRST_TIME),'Mon DD') "DG Date",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'00',1,0)),'9999')
"12AM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'01',1,0)),'9999')
"01AM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'02',1,0)),'9999')
"02AM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'03',1,0)),'9999')
"03AM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'04',1,0)),'9999')
"04AM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'05',1,0)),'9999')
"05AM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'06',1,0)),'9999')
"06AM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'07',1,0)),'9999')
"07AM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'08',1,0)),'9999')
"08AM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'09',1,0)),'9999')
"09AM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'10',1,0)),'9999')
"10AM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'11',1,0)),'9999')
"11AM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'12',1,0)),'9999')
"12PM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'13',1,0)),'9999')
"1PM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'14',1,0)),'9999')
"2PM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'15',1,0)),'9999')
"3PM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'16',1,0)),'9999')
"4PM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'17',1,0)),'9999')
"5PM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'18',1,0)),'9999')
"6PM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'19',1,0)),'9999')
"7PM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'20',1,0)),'9999')
"8PM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'21',1,0)),'9999')
"9PM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'22',1,0)),'9999')
"10PM",
TO_CHAR(SUM(DECODE(TO_CHAR(FIRST_TIME,'HH24'),'23',1,0)),'9999')
"11PM"
FROM V$LOG_HISTORY
GROUP BY TRUNC(FIRST_TIME)
ORDER BY TRUNC(FIRST_TIME) DESC
/
```

Script 2:

```
SELECT created, COUNT(*) AS archives
FROM dba_data_archives
GROUP BY TO_CHAR(created, 'YYYY-MM-DD HH24')
ORDER BY created;
```

8. Find Active Transactions in DB:

Script 1:

```
col name format a10
col username format a8
col osuser format a8
col start_time format a17
col status format a12
tti 'Active transactions'
select s.sid,username,t.start_time, r.name, t.used_ublk "USED
BLKS",
decode(t.space, 'YES', 'SPACE TX',
decode(t.recursive, 'YES', 'RECURSIVE TX',
decode(t.noundo, 'YES', 'NO UNDO TX', t.status)
)) status
from sys.v_$transaction t, sys.v_$rollname r, sys.v_$session s
where t.xidusn = r.usn
and t.ses_addr = s.saddr
/
```

Script 2:

```
SELECT * FROM v$session WHERE status = 'ACTIVE';
```

9. Find Who Locked Your Account:

Script 1:

```
-- Return code 1017 ( INVALID LOGIN ATTEMPT)
-- Return code 28000 ( ACCOUNT LOCKED)
set pagesize 1299
set lines 299
col username for a15
col userhost for a13
col timestamp for a39
col terminal for a23
SELECT username,userhost,terminal,timestamp,returncode
FROM dba_audit_session
WHERE username='&USER_NAME' and returncode in (1017,28000);
```

Script 2:

```
SELECT s.username, l.session_id, l.blocking_session_status
FROM v$session l, v$session s
WHERE l.blocking_session_status = 'ACTIVE'
  AND l.blocking_session_status = s.sid;
```

10. Find Duplicate Rows in Table:

Script 1:

```
--- Reference metalink id - 332494.1
-- Save as duplicate.sql and run as @duplicate.sql
REM This is an example SQL*Plus Script to detect duplicate rows
from
REM a table.
REM
set echo off
set verify off heading off
undefine t
undefine c
prompt
```

```
prompt
prompt Enter name of table with duplicate rows
prompt
accept t prompt 'Table: '
prompt
select 'Table '||upper('&&t') from dual;
describe &&t
prompt
prompt Enter name(s) of column(s) which should be unique. If
more than
prompt one column is specified, you MUST separate with commas.
prompt
accept c prompt 'Column(s): '
prompt
select &&c from &&t
where rowid not in (select min(rowid) from &&t group by &&c)
/
```

Script 2:

```
SELECT * FROM your_table
GROUP BY column1, column2, ...
HAVING COUNT(*) > 1;
```

11. Generate Resize Datafile Script:

Script 1:

```
--generate resize datafile script without ORA-03297 error
select 'alter database datafile'||' '''||file_name||''''||'
resize '||round(highwater+2)||' '||'m'||';' from (
select /*+ rule */
a.tablespace_name,
a.file_name,
a.bytes/1024/1024 file_size_MB,
(b.maximum+c.blocks-1)*d.db_block_size/1024/1024 highwater
from dba_data_files a         ,
(select file_id,max(block_id) maximum
from dba_extents
group by file_id) b,
dba_extents c,
(select value db_block_size
from v$parameter
where name='db_block_size') d
where a.file_id=  b.file_id
and   c.file_id  = b.file_id
and   c.block_id = b.maximum
order by a.tablespace_name,a.file_name);
```

Script 2:

```
ALTER DATABASE DATAFILE '<file_name>' RESIZE <new_size>M;
```

12. Database Growth per Month:

Script 1:

```
select to_char(creation_time, 'MM-RRRR') "Month",
sum(bytes)/1024/1024/1024 "Growth in GB
from sys.v_$datafile
where to_char(creation_time,'RRRR')='&YEAR_IN_YYYY_FORMAT'
group by to_char(creation_time, 'MM-RRRR')
order by to_char(creation_time, 'MM-RRRR');
```

Script 2:

```
SELECT TO_CHAR(snap_time, 'YYYY-MM') AS month,
       (used_space - (LAG(used_space) OVER (ORDER BY
snap_time)))/1024/1024/1024 AS growth_gb
FROM dba_hist_snapshot_stat
ORDER BY snap_time;
```

13. Get Database Uptime:

Script 1:

```
select to_char(startup_time, 'DD-MM-YYYY
HH24:MI:SS'),floor(sysdate-startup_time) DAYS from v$Instance;
```

Script 2:

```
SELECT SYSTIMESTAMP - startup_time AS uptime FROM v$instance;
```

14. SCN to Timestamp and Vice Versa:

Script 1:

```
-- Get current scn value:
select current_scn from v$database;
-- Get scn value at particular time:
select timestamp_to_scn('19-JAN-24:22:00:10') from dual;
-- Get timestamp from scn:
select scn_to_timestamp(224292)from dual;
```

Script 2:

```
SELECT TO_CHAR(dbms_utility.SCN_TO_TIMESTAMP(<scn_value>),
'YYYY-MM-DD HH24:MI:SS') AS timestamp;

SELECT dbms_utility.SCN_TO_NUMBER(TO_CHAR(SYSDATE,
'YYYYMMDDHH24MISS')) AS scn;
```

15. Disable/Enable All Triggers of Schema:

Script:

```
-- Disable
BEGIN
  FOR t IN (SELECT * FROM user_triggers) LOOP
    EXECUTE DBMS_METADATA.DISABLE_TRIGGER(t.owner,
t.trigger_name);
  END LOOP;
END;
/

-- Enable
BEGIN
  FOR t IN (SELECT * FROM user_triggers) LOOP
    EXECUTE DBMS_METADATA.ENABLE_TRIGGER(t.owner,
t.trigger_name);
```

```
  END LOOP;
END;
/
```

16. Get Row Count of All Tables in a Schema:

Script 1:

```
select table_name,
to_number(extractvalue(dbms_xmlgen.getXMLtype('select /*+
PARALLEL(8) */ count(*) cnt from
"&&SCHEMA_NAME".'||table_name),'/ROWSET/ROW/CNT'))
rows_in_table from dba_TABLES
where owner='&&SCHEMA_NAME';
```

Script 2:

```
SELECT table_name, COUNT(*) AS row_count
FROM user_tables
GROUP BY table_name;
```

17. Monitor Index Usage:

Script 1:

```
--Index monitoring is required, to find whether indexes are
really in use or not. Unused can be dropped to avoid overhead.
--First enable monitoring usage for the indexes.
alter index siebel.S_ASSET_TEST monitoring usage;
--Below query to find the index usage:
select * from v$object_usage;
```

Script 2:

```
SELECT owner, table_name, index_name,
       DECODE(latency, 0, 'NA', latency) AS avg_latency,
       requests
FROM dba_indexes;
```

18. Spool SQL Query Output to HTML:

Script 1:

```
--We can spool output of an sql query to html format:
set pages 5000
SET MARKUP HTML ON SPOOL ON PREFORMAT OFF ENTMAP ON -
HEAD "<TITLE>EMPLOYEE REPORT</TITLE> -
<STYLE type='text/css'> -
<!-- BODY {background: #FFFFC6} --> -
</STYLE>" -
BODY "TEXT='#FF00Ff'" -
TABLE "WIDTH='90%' BORDER='5'"
spool report.html
Select * from scott.emp;
spool off
exit
```

Script 2:

```
SET LINESIZE 300
SET PAGESIZE 0
SPOOL your_file.html
START HTML ... (HTML formatting commands) ...
SELECT * FROM your_table;
... (More HTML formatting commands) ...
SPOOL OFF;
```

19. Get Installed SQL Patches in DB:

Script 1:

```
--- From 12c onward
set lines 2000
select patch_id,status,description from dba_registry_sqlpatch;
```

Script 2:

```
-- Oracle:
SELECT * FROM dba_registry WHERE KEY LIKE '%Patch%' OR KEY LIKE
'%Bundle%';
```

```
-- SQL Server:
SELECT * FROM sys.sql_modules m
INNER JOIN sys.extended_properties ep ON m.object_id =
ep.major_id
WHERE ep.name = N'MS_SQLServer_PatchLevel';
```

Optimization: Filter results based on keywords like "Patch" or "Bundle" in Oracle to improve efficiency.

20. Cleanup Orphaned Datapump Jobs:

Script 1:

```
-- Find the orphaned Data Pump jobs:
SELECT owner_name, job_name, rtrim(operation) "OPERATION",
rtrim(job_mode) "JOB_MODE", state, attached_sessions
FROM dba_datapump_jobs
WHERE job_name NOT LIKE 'BIN$%' and state='NOT RUNNING'
ORDER BY 1,2;
-- Drop the tables
SELECT 'drop table ' || owner_name || '.' || job_name || ';'
FROM dba_datapump_jobs WHERE state='NOT RUNNING' and job_name
NOT LIKE 'BIN$%'
```

Script 2:

```
-- Oracle:
DECLARE
  CURSOR c_orphan_jobs IS
    SELECT job_name
    FROM dba_datapump_jobs
    WHERE status NOT IN ('SUCCEEDED', 'FAILED', 'STOPPED');
BEGIN
  FOR rec IN c_orphan_jobs LOOP
    DBMS_SCHEDULER.DROP_JOB(job_name => rec.job_name);
  END LOOP;
END;
/

-- SQL Server:
-- Use Management Studio or PowerShell cmdlets for cleanup.
```

21. Installed RDBMS Components:

Script 1:

```
col comp_id for a10
col comp_name for a56
col version for a12
col status for a10
set pagesize 200
set lines 200
set long 999
select comp_id,comp_name,version,status from dba_registry;
```

Script 2:

```
-- Oracle:
SELECT * FROM dba_registry WHERE KEY LIKE '%Component%';

-- SQL Server:
SELECT SERVERPROPERTY('ProductVersion'),
SERVERPROPERTY('ProductVersionFull')
```

Optimization: Filter results based on keywords like "Component" in Oracle to improve efficiency.

22. Characterset Info of Database:

Script 1:

```
set pagesize 200
set lines 200
select parameter,value from v$nls_parameters where parameter
like 'NLS_%CHAR%';
```

Script 2:

```
-- Oracle:
SELECT * FROM nls_database_parameters;

-- SQL Server:
SELECT SERVERPROPERTY('Collation_Name')
```

23. View/Modify AWR Retention:

Script 1:

```
-- View current AWR retention period
select retention from dba_hist_wr_control;
-- Modify retention period to 7 days and interval to 30 min
select dbms_workload_repository.modify_snapshot_settings
(interval => 30, retention => 10080);
NOTE - 7 DAYS = 7*24*3600= 10080 minutes
```

Script 2:

```
-- Oracle:
SELECT * FROM dba_retention_defs WHERE retention_name = 'AWR';

-- SQL Server:
-- Use Management Studio or PowerShell cmdlets for
configuration.
```

24. Find Optimal Undo Retention Size:

Script 1:

```
SELECT d.undo_size / (1024 * 1024) "ACTUAL UNDO SIZE [MByte]",
SUBSTR(e.value, 1, 25) "UNDO RETENTION [Sec]",
(TO_NUMBER(e.value) * TO_NUMBER(f.value) * g.undo_block_per_sec)
/
(1024 * 1024) "NEEDED UNDO SIZE [MByte]"
FROM (SELECT SUM(a.bytes) undo_size
FROM gv$datafile a, gv$tablespace b, dba_tablespaces c
WHERE c.contents = 'UNDO'
AND c.status = 'ONLINE'
AND b.name = c.tablespace_name
AND a.ts# = b.ts#) d,
gv$parameter e,
gv$parameter f,
(SELECT MAX(undoblks / ((end_time - begin_time) * 3600 * 24))
undo_block_per_sec
FROM v$undostat) g
```

```
WHERE e.name = 'undo_retention'
AND f.name = 'db_block_size';
```

Script 2:

```
-- Oracle:
SELECT DBMS_UNDO.ESTIMATE_UNDO_RETENTION_SIZE(<retention_days>)
AS est_size_mb
FROM DUAL;

-- SQL Server:
-- Use built-in tools or recommendations from Microsoft.
```

25. Purge Old AWR Snapshots:

Script 1:

```
-- Find the AWR snapshot details.
select snap_id,begin_interval_time,end_interval_time from
sys.wrm$_snapshot order by snap_id
-- Purge snapshot between snapid 612 to 700
execute dbms_workload_repository.drop_snapshot_range(low_snap_id
=>612 , high_snap_id =>700);
-- Verify again
select snap_id,begin_interval_time,end_interval_time from
sys.wrm$_snapshot order by snap_id
```

Script 2:

```
-- Oracle:
DBMS_SNAPSHOT.DROP(<snapshot_id>);

-- SQL Server:
-- Use Management Studio or PowerShell cmdlets for cleanup.
```

26. Modify Moving Window Baseline Size:

Script 1:

```
-- Check the current moving window baseline size:
select BASELINE_TYPE,MOVING_WINDOW_SIZE from dba_hist_baseline;
-- Modify window_size to (7 days):
```

```
execute
dbms_workload_repository.modify_baseline_window_size(window_size
=> 7);
```

Script 2:

```
-- Oracle:
ALTER SYSTEM SET "_optimizer_mvwb_size" = <new_size>;

-- SQL Server:
-- Use Management Studio or PowerShell cmdlets for
configuration.
```

27. Open Database Link Information:

Script 1:

```
set pagesize 200
set lines 200
col db_link for a19
set long 999
SELECT db_link,
owner_id,
logged_on,
heterogeneous,
open_cursors,
in_transaction,
update_sent
FROM gv$dblink
ORDER BY db_link;
```

Script 2:

```
-- Oracle:
SELECT * FROM dba_db_links;

-- SQL Server:
SELECT * FROM sys.linked_servers;
```

28. Utilization of Current Redo Log (in %):

Script 1:

```
SELECT le.leseq "Current log sequence No",
100*cp.cpodr_bno/le.lesiz "Percent Full",
cp.cpodr_bno "Current Block No",
le.lesiz "Size of Log in Blocks"
FROM x$kcccp cp, x$kccle le
WHERE le.leseq =CP.cpodr_seq
AND bitand(le.leflg,24) = 8
/
```

Script 2:

```
-- Oracle:
SELECT (NVL(SUM(CASE WHEN archived = 'NO' THEN bytes ELSE 0
END), 0) / SUM(bytes)) * 100 AS pct_used
FROM v$logfile;

-- SQL Server:
-- Use DMV (Dynamic Management Views) for monitoring.
```

29. Table Not Having Index on FK Column:

Script 1:

```
select * from (
select c.table_name, co.column_name, co.position column_position
from user_constraints c, user_cons_columns co
where c.constraint_name = co.constraint_name
and c.constraint_type = 'R'
minus
select ui.table_name, uic.column_name, uic.column_position
from user_indexes ui, user_ind_columns uic
where ui.index_name = uic.index_name
)
order by table_name, column_position;
select
a.constraint_name cons_name
,a.table_name tab_name
,b.column_name cons_column
,nvl(c.column_name,'***No Index***') ind_column
from user_constraints a
join
```

```
user_cons_columns b on a.constraint_name = b.constraint_name
left outer join
user_ind_columns c on b.column_name = c.column_name
and b.table_name = c.table_name
where constraint_type = 'R'
order by 2,1;
```

Script 2:

```
-- Oracle:
SELECT tc.table_name, c.column_name
FROM user_tables tc
JOIN user_constraints uc ON tc.table_name = uc.table_name
```

30. Get CPU and Memory Info of DB Server:

Script 1:

```
set pagesize 200
set lines 200
col name for a21
col stat_name for a25
col value for a13
col comments for a56
select STAT_NAME,to_char(VALUE) as VALUE ,COMMENTS from v$osstat
where
stat_name IN ('NUM_CPUS','NUM_CPU_CORES','NUM_CPU_SOCKETS')
union
select STAT_NAME,VALUE/1024/1024/1024 || ' GB' ,COMMENTS from
v$osstat where stat_name IN ('PHYSICAL_MEMORY_BYTES');
```

Script 2:

```
SELECT * FROM v$resource_pool_wait_stat; -- CPU utilization
SELECT * FROM v$resource_pool_memory_usage; -- Memory usage

SELECT * FROM sys.dm_os_cpu_busy -- CPU utilization
SELECT * FROM sys.dm_os_memory_clerks -- Memory usage
```

Optimizations:

- Utilize built-in views or functions designed for performance monitoring.

- Consider using management tools provided by the database vendor for a more comprehensive overview.

31. Get Database Incarnation Info:

Script 1:

```
set heading off
set feedback off
select 'Incarnation Destination Configuration' from dual;
select '*************************************' from dual;
set heading on
set feedback on
select INCARNATION# INC#, RESETLOGS_CHANGE# RS_CHANGE#,
RESETLOGS_TIME,
PRIOR_RESETLOGS_CHANGE# PRIOR_RS_CHANGE#, STATUS,
FLASHBACK_DATABASE_ALLOWED FB_OK from v$database_incarnation;
```

Script 2:

```
-- Oracle:
SELECT * FROM v$system_event WHERE name = 'instance
incarnation';

-- SQL Server:
SELECT SERVERPROPERTY('ServerVersionInfo') -- Information
includes incarnation ID
```

32. View Timezone Info in DB:

Script 1:

```
SELECT version FROM v$timezone_file;
SELECT PROPERTY_NAME, SUBSTR(property_value, 1, 30) value
FROM DATABASE_PROPERTIES
WHERE PROPERTY_NAME LIKE 'DST_%'
ORDER BY PROPERTY_NAME;
```

Script 2:

```
-- Oracle:
SELECT * FROM nls_database_parameters WHERE parameter =
'TIME_ZONE';

-- SQL Server:
SELECT @@SERVERPROPERTY('timezoneoffset') -- Offset from UTC in
minutes
```

Additional Considerations:

- **Security:** Ensure you have the necessary privileges to execute the provided queries.
- **Database version:** The specific syntax and available views/functions might differ slightly between database versions.
- **Alternative methods:** Explore management tools or vendor-specific utilities for comprehensive performance and configuration information.

By following these guidelines and considering optimization techniques, you can efficiently retrieve the desired database information while minimizing resource consumption.