Database Practices- DB 12c
Oracle Banking Payments
Release 14.2.0.0.0
[November] 2018

ORACLE
FINANCIAL SERVICES

Table of Contents

1. Database Initialization Parameters

Oracle Banking standard database initialization parameters have been derived after performing the required benchmark tests (Performance Load tests).

Note: Since some of the initialization parameters values are specific to customer volume, parameters should be derived using **FCPAYMENTS-Disk-Layouts-initparams-12c-version12.xlsm** excel sheet base lined along with this document.

Following are the Parameters with the details and its relevance to Banking:

1.1 DB_WRITER_PROCESSES

This is useful for systems that modify data heavily. It specifies the initial number of database writer processes for an instance.

Property	Description
Parameter type	Integer
Default value	1 or CPU_COUNT / 8, whichever is greater
Range of Values	1 to 20
Recommended	Refer **FCPAYMENTS-Disk-Layouts-initparams-12c-version12.xlsm**

Oracle Banking relevance

Multiple DB writer processes helps faster flushing of data to disk. To arrive right value, refer **FCPAYMENTS-Disk-Layouts-initparams-12c-version12.xlsm** excel.

1.2 CURSOR_SHARING

Determines what kind of SQL statements can share the same cursors.

Property	Description
Parameter type	String
Default value	EXACT
Recommended Value	Force

Oracle Banking relevance

Some of the Banking sql statements are generated dynamically. So they contain literal values in the WHERE clause conditions. This results in large numbers of nearly identical statements with separate parse trees in Oracle's library cache, which can slow performance and cause latch problems.

By setting cursor_sharing to FORCE database convert literals to bind variables before parsing the statement.

1.3 DB_CACHE_ADVICE

This enables or disables statistics gathering used for predicting behavior with different cache sizes through the V$DB_CACHE_ADVICE performance view.

Property	Description
Parameter type	String
Syntax	DB_CACHE_ADVICE = { ON | READY | OFF }
Default value	If STATISTICS_LEVEL is set to TYPICAL / ALL, then ON If STATISTICS_LEVEL is set to BASIC, then OFF
Recommended Value	OFF (Should be ON while Performance Monitoring)

Oracle Banking relevance

Turning ON advisory will have an extra overhead. Please note it should be ON, only during performance monitoring.

1.4 FAST_START_MTTR_TARGET

This enables you to specify the number of seconds the database takes to perform crash recovery of a single instance. When specified, FAST_START_MTTR_TARGET is overridden by LOG_CHECKPOINT_INTERVAL.

Property	Description
Parameter type	Integer
Default value	0
Range of values	0 to 3600 seconds
Recommended Values	300

Oracle Banking relevance

If FAST_START_MTTR_TARGET is not set to 300 then run time performance for write/redo generation intensive workloads will not be optimized. This will reduce checkpoint writes from DBWR processes, making more room for LGWR IO. To optimize run time performance for write/redo generation intensive workloads, increase the FAST_START_MTTR_TARGET initialization parameter to 300.

1.5 FILESYSTEM_IO_OPTIONS

This specifies the IO operation for file system files.

Property	Description
Parameter type	String
Default value	There is no default value
Range of values	NONE,SETALL,DIRECTIO,ASYNC
Recommended Values	SETALL

Oracle Banking relevance

By setting parameter value to SETALL, Oracle can take advantage of direct I/O and asynchronous I/O on supported platforms. However, this parameter will not have any effect if ASM is being used.

1.6 JOB_QUEUE_PROCESSES

This specifies the maximum number of processes that can be created for the execution of jobs. It specifies the number of job queue processes per instance (J000, J999).

Property	Description
Parameter type	Integer
Default value	0
Range of values	0 to 1000
Recommended Values	Refer **FCPAYMENTS-Disk-Layouts-initparams-12c-version12.xlsm**

Oracle Banking relevance

This parameter has to be set with respect to the maximum number of jobs (dbms_jobs). To arrive at the right value, refer FCPAYMENTS-Disk-Layouts-initparams-12c-version12.xlsm excel.

1.7 NLS_DATE_FORMAT

This specifies the default date format to use with the TO_CHAR and TO_DATE functions.

Property	Description
Parameter type	String
Syntax	NLS_DATE_FORMAT = "format"
Default value	Derived from NLS_TERRITORY
Recommended Values	DD-MON-RRRR

Oracle Banking relevance

Banking standard date format.

1.8 OPEN_CURSORS

This specifies the maximum number of open cursors (handles to private SQL areas) a session can have at once. You can use this parameter to prevent a session from opening an excessive number of cursors.

Property	Description
Parameter type	Integer
Default value	50
Modifiable	ALTER SYSTEM
Range of values	1 to 4294967295 (4 GB -1)
Recommended Values	5000

Oracle Banking relevance

It is important to set the value of OPEN_CURSORS high enough to prevent FCPAYMENTS application from running out of open cursors (ORA-01000: maximum open cursors exceeded).

1.9 OPTIMIZER_DYNAMIC_SAMPLING

This controls the level of dynamic sampling performed by the optimizer.

Property	Description
Parameter type	Integer
Default value	If OPTIMIZER_FEATURES_ENABLE is set to 10.0.0 or higher, then 2 If OPTIMIZER_FEATURES_ENABLE is set to 9.2.0, then 1 If OPTIMIZER_FEATURES_ENABLE is set to 9.0.1 or

	lower, then 0
Recommended Values	1
Range of values	0 to 10

Oracle Banking relevance

Dynamic Sampling is a method of gathering additional statistics during optimization by recursively sampling statements. When dynamic sampling is enabled, queries are recursively generated by Oracle to test various selectivity based upon real values in order to improve their accuracy. This can result in the production of better explain plans.

Value 1 Sample all tables that have not been analyzed that meet certain criteria.

1.10 OPTIMIZER_INDEX_CACHING

This lets you adjust the behavior of cost-based optimization to favor nested loops joins and IN-list iterators.

Property	Description
Parameter type	Integer
Default value	0
Recommended Values	90
Range of values	0 to 100

Oracle Banking relevance

The cost of executing an index using IN-list iterators or of executing nested loops join when an index is used to access the inner table depends on the caching of that index in the buffer cache. Banking favors nested loop joins by setting optimizer_index_caching to 90.

1.11 OPTIMIZER_INDEX_COST_ADJ

This lets you tune optimizer behavior for access path selection to be more or less index friendly - that is, to make the optimizer more or less prone to selecting an index access path over a full table scan.

Property	Description
Parameter type	Integer
Default value	100
Recommended Values	50
Range of values	1 to 10000

Oracle Banking relevance

Banking favors index read over full table scan as it is very useful when optimizer favors to give a lower cost to index scans over full-table scans.

1.12 PLSQL_CODE_TYPE

This specifies the compilation mode of the PL/SQL units.

Property	Description
Parameter type	String
Default value	INTERPRETED
Recommended values	NATIVE
Range of values	INTERPRETED, NATIVE

Oracle Banking relevance

The PL SQL interpreter overhead will be minimal when set to NATIVE.

1.13 PLSQL_OPTIMIZE_LEVEL

This specifies the optimization level that will be used to compile PL/SQL library units. The higher the setting of this parameter, the more effort the compiler makes to optimize PL/SQL library units.

Property	Description
Parameter type	Integer
Default value	2
Recommended values	3
Range of values	0 to 3

Oracle Banking relevance

This applies a wide range of modern optimization techniques beyond those of level 1 including changes which may move source code relatively far from its original location.

1.14 PROCESSES

This specifies the maximum number of operating system user processes that can simultaneously connect to Oracle. Its value should allow for all background processes such as locks, job queue processes, and parallel execution processes.

Property	Description

Parameter type	Integer
Default value	100
Range of values	6 to operating system dependent
Recommended values	Refer **FCPAYMENTS-Disk-Layouts-initparams-12c-version12.xlsm**

Oracle Banking relevance

This parameter can set be set with respect to maximum no of sessions connected to DB.

1.15 QUERY_REWRITE_ENABLED

Allows you to enable or disable query rewriting globally for the database.

Property	Description
Parameter type	String
Syntax	QUERY_REWRITE_ENABLED = { false | true | force }
Default value	If OPTIMIZER_FEATURES_ENABLE is set to 10.0.0 or higher, then true If OPTIMIZER_FEATURES_ENABLE is set to 9.2.0 or lower, then false
Recommended values	FALSE

Oracle Banking relevance

Banking doesn't uses function-based indexes.

1.16 REMOTE_DEPENDENCIES_MODE

Specifies how Oracle should handle dependencies upon remote PL/SQL stored procedures.

Property	Description
Parameter type	String
Syntax	REMOTE_DEPENDENCIES_MODE = { TIMESTAMP | SIGNATURE }
Default value	TIMESTAMP
Recommended values	SIGNATURE

Oracle Banking relevance

Oracle allows the procedure to execute as long as the signatures are considered safe. This setting allows client PL/SQL applications to be run without recompilation.

1.17 RESULT_CACHE_MAX_SIZE

RESULT_CACHE_MAX_SIZE specifies the maximum amount of SGA memory (in bytes) that can be used by the Result Cache.

Property	Description
Parameter type	Big integer
Syntax	`RESULT_CACHE_MAX_SIZE = integer [K | M | G]`
Default value	Derived from the values of `SHARED_POOL_SIZE`, `SGA_TARGET`, and `MEMORY_TARGET`
Recommended values	0.5% of SGA

Oracle Banking relevance

■Automatic memory management

If you are using the MEMORY_TARGET initialization parameter to specify memory allocation, Oracle Database allocates 0.25% of the value of the MEMORY_TARGET parameter to the result cache.

■Automatic shared memory management

If you are managing the size of the shared pool using the SGA_TARGET initialization parameter, Oracle Database allocates 0.50% of the value of the SGA_TARGET parameter to the result cache.

■Manual shared memory management

If you are managing the size of the shared pool using the SHARED_POOL_SIZE initialization parameter, then Oracle Database allocates 1% of the shared pool size to the result cache.

1.18 CLIENT_RESULT_CACHE_LAG

CLIENT_RESULT_CACHE_LAG specifies the maximum time (in milliseconds) since the last round trip to the server, before which the OCI client query execute makes a round trip to get any database changes related to the queries cached on the client.

Property	Description
Parameter type	Big integer

Syntax	`CLIENT_RESULT_CACHE_LAG = integer`
Default value	3000
Recommended values	10,800,000(3 hours)

1.19 CLIENT_RESULT_CACHE_SIZE

CLIENT_RESULT_CACHE_SIZE specifies the maximum size of the client per-process result set cache (in bytes). All OCI client processes inherit this maximum size. Setting a nonzero value enables the client query cache feature. This can be overridden by the client configuration parameter OCI_RESULT_CACHE_MAX_SIZE

Property	**Description**
Parameter type	Big integer
Syntax	`CLIENT_RESULT_CACHE_SIZE=integer [K|M|G]`
Default value	0
Recommended values	32K

1.20 SESSION_CACHED_CURSORS

Specifies the number of session cursors to cache. Repeated parse calls of the same SQL statement cause the session cursor for that statement to be moved into the session cursor cache. Subsequent parse calls will find the cursor in the cache and do not need to reopen the cursor. Oracle uses a least recently used algorithm to remove entries in the session cursor cache to make room for new entries when needed.

Property	**Description**
Parameter type	Integer
Default value	50
Recommended values	400
Range of values	0 to operating system-dependent

Oracle Banking relevance

This helps to cache the cursor thus avoid parsing of the cursor which heavy CPU intensive particularly in batch.

1.21 SKIP_UNUSABLE_INDEXES

Enables or disables the use and reporting of tables with unusable indexes or index partitions.

Property	Description
Parameter type	Boolean
Default value	true
Recommended values	FALSE
Range of values	true / false

Oracle Banking relevance

TRUE enables error reporting of indexes marked UNUSABLE. This setting does not allow inserts, deletes, and updates on tables with unusable indexes or index partitions. IT is set to false because Banking application should throw error if any of the indexes become UNUSABLE.

1.22 UNDO_RETENTION

This specifies (in seconds) the low threshold value of undo retention. For AUTOEXTEND undo tablespaces, the system retains undo for at least the time specified in this parameter, and automatically tunes the undo retention period to satisfy the undo requirements of the queries. For fixed- size undo tablespaces, the system automatically tunes for the maximum possible undo retention period, based on undo tablespace size and usage history, and ignores UNDO_RETENTION unless retention guarantee is enabled.

The UNDO_RETENTION parameter can only be honored if the current undo tablespace has enough space. If an active transaction requires undo space and the undo tablespace does not have available space, then the system starts reusing unexpired undo space. This action can potentially cause some queries to fail with a "snapshot too old" message.

Property	Description
Parameter type	Integer
Default value	900
Range of values	0 to 231 – 1
Recommended values	1800

Oracle Banking relevance

Increased value along with automatic undo management helps to avoid “snapshot too old error”.

1.23 UTL_FILE_DIR

Lets you specify one or more directories that Oracle should use for PL/SQL file I/O. If you are specifying multiple directories, you must repeat the UTL_FILE_DIR parameter for each directory on separate lines of the initialization parameter file.

Property	Description
Parameter type	String
Syntax	UTL_FILE_DIR = pathname
Default value	There is no default value
Recommended values	/tmp/Banking

Oracle Banking relevance

Security recommends to create one single directory for writing all the DEBUG related files. This should be in sync with the DEBUG related parameter values mentioned in CSTB_PARAM table. If any other components require to write to a different location, the same needs to be updated in UTL_FILE_DIR as well.

1.24 DEFERRED_SEGMENT_CREATION

Specifies the semantics of deferred segment creation. If set to true, then segments for non-partitioned tables and their dependent objects (LOBs, indexes) will not be created until the first row is inserted into the table.

Property	Description
Parameter type	Boolean
Default value	True
Modifiable	ALTER SESSION, ALTER SYSTEM
Recommended value	False

Oracle Banking relevance

All the Banking tables should be imported / created even though there is no record in the table.

1.25 LOG_BUFFER

Recommended Value: 100M

Oracle Banking relevance

The default log buffer size is too small as Banking performs heavy DML during batch processing.

1.26 _ALLOW_LEVEL_WITHOUT_CONNECT_BY

Recommended Value: TRUE

This parameter is set to avoid following error,

- After Upgrading To Oracle 10g, Getting ORA-01788 When Running A Query That Includes The LEVEL Pseudo Column [ID 455953.1]

1.27 PGA_AGGREGATE_LIMIT

Recommended Value: 0

Oracle Banking Relevance:

Setting this parameter limits the pga consumed by the instance, hence might cause failure to few of the running processes.

1.28 _optimizer_adaptive_features

Property	Description
Parameter type	Boolean
Default value	True
Modifiable	ALTER SESSION, ALTER SYSTEM
Recommended value	False

Oracle Banking relevance

Some of the Banking sql statements are generated dynamically. So they contain literal values in the WHERE clause conditions. This results in large numbers of nearly identical statements with separate parse trees in Oracle's library cache, which can slow performance and cause latch problems. Also this can lead to a wrong explain plan.

2. Redo Log Files

The default redo log files groups and size is inadequate to run Banking. Hence, the recommended are:

- 6 redo log groups
- Redo log file size
  - 1 GB each for the DB size up to 1 TB
  - 2 GB each for DB size more than 1 TB

3. Tablespace Layout and Moving Tables to Respective Tablespaces

Oracle Banking tables and indexes are placed in corresponding tablespaces according to their usage. I.e. heavily populated tables and corresponding indexes are placed in tablespaces with higher extent size. Whereas the maintenance tables where the data population is less will be placed in a tablespace with smaller extent size. This avoids frequent space allocation in turn improve the performance.

For example table ACTB_HISTORY is heavily populated. So this table and its indexes will be placed in tablespace FCCDATAXL and FCCINDXXL respectively where extent size is high. The table STTM_BRANCH and its indexes are placed in tablespace FCCDATASML and FCCINDXSML respectively which is having smaller extent size.

Oracle Banking Standard Tablespaces are as follows,

Tablespace name	Tablespace type	Extent management	Segment space management
FCCDATASML	DATA	LOCAL	AUTO
FCCINDXSML	INDEX	LOCAL	AUTO
FCCDATAMED	DATA	LOCAL	AUTO
FCCINDXMED	INDEX	LOCAL	AUTO
FCCDATALAR	DATA	LOCAL	AUTO
FCCINDXLAR	INDEX	LOCAL	AUTO
FCCDATAXL	DATA	LOCAL	AUTO
FCCINDXXL	INDEX	LOCAL	AUTO
FCCDFLT	AD HOC	LOCAL	AUTO

Note: Tablespaces extent size depends on the Banking implementation (i.e. Small, Medium and Large). So these parameters are to be derived using base lined excel **FCPAYMENTS-Disk-Layouts-initparams-12c-version12.xlsm** based on implementation.

For the table to tablespace mapping, refer base lined excel sheet FCPAYMENTS-Tablespace-Distribution_v12.xlsx.

Sample script to move table and index:

```
Alter table STTM_CUST_ACCOUNT_DORMANCY move tablespace FCCDATALAR;

Alter index IND_DRREF rebuild tablespace FCCINDXLAR;
```

Similarly all tables and indexes should be moved to respective tablespaces.

4. Table & Index Partitioning

Table and index partitioning helps to reduce the contention and GC related delays in RAC environment. Table and index partitioning is mandatory if you have deployed Oracle Banking in RAC database.

Following are the list of tables to be partitioned:

NAME	PARTITIONING_TYPE	COLUMN_NAME
ACTB_ACCBAL_HISTORY	LIST	BRANCH_CODE
ACTB_DAILY_LOG	LIST	AC_BRANCH
ACTB_HISTORY	LIST	AC_BRANCH
ACTB_MONTHLY_TOV_HIST	LIST	BRANCH_CODE
ACTB_VD_BAL	LIST	BRN
CATM_CHECK_BOOK	LIST	BRANCH
CATM_CHECK_DETAILS	LIST	BRANCH
CLTB_ACCOUNT_APPS_MASTER	RANGE	BRANCH_CODE,PROCESS_NO
CLTB_ACCOUNT_COMPONENTS	LIST	BRANCH_CODE
CLTB_ACCOUNT_COMP_BALANCES	LIST	BRANCH_CODE
CLTB_ACCOUNT_COMP_BAL_BREAKUP	LIST	BRANCH_CODE
CLTB_ACCOUNT_COMP_BAL_SUMMARY	LIST	BRANCH_CODE
CLTB_ACCOUNT_COMP_CALC	LIST	BRANCH_CODE
CLTB_ACCOUNT_COMP_SCH	LIST	BRANCH_CODE
CLTB_ACCOUNT_EVENTS_ADVICES	LIST	BRANCH_CODE
CLTB_ACCOUNT_EVENTS_DIARY	RANGE	BRANCH_CODE,PROCESS_NO
CLTB_ACCOUNT_PARTIES	LIST	BRANCH_CODE
CLTB_ACCOUNT_ROLL_COMP	LIST	BRANCH_CODE
CLTB_ACCOUNT_SCHEDULES	LIST	BRANCH_CODE
CLTB_ACCOUNT_UDE_EFF_DATES	LIST	BRANCH_CODE
CLTB_ACCOUNT_UDE_VALUES	LIST	BRANCH_CODE
CLTB_ACC_COMPOUNDING_DATES	LIST	BRANCH_CODE
CLTB_AMOUNT_PAID	LIST	BRANCH_CODE
CLTB_AMOUNT_PAID_HISTORY	LIST	BRANCH_CODE
CLTB_AMOUNT_RECD	LIST	BRANCH_CODE
CLTB_CALC_DATES	LIST	BRANCH_CODE
CLTB_DISBR_SCHEDULES	LIST	BRANCH_CODE
CLTB_EVENT_ENTRIES	RANGE	BRANCH_CODE,PROCESS_NO
CLTB_EVENT_ENTRIES_PENDING	LIST	BRANCH_CODE
CLTB_EVENT_REMARKS	LIST	BRANCH_CODE
CLTB_LIQ	LIST	BRANCH_CODE
CLTB_RECON	LIST	BRANCH_CODE
CLTB_REVISION_ACCOUNTS	LIST	BRANCH_CODE
CLTB_REVN_SCHEDULES	LIST	BRANCH_CODE

CLTP_ACCOUNT_COMP_BALANCES	LIST	BRANCH_CODE
CLTP_ACCOUNT_COMP_CALC	LIST	BRANCH_CODE
CLTP_ACCOUNT_COMP_SCH	LIST	BRANCH_CODE
CLTP_ACCOUNT_SCHEDULES	LIST	BRANCH_CODE
CSTB_AMOUNT_DUE	HASH	CONTRACT_REF_NO
CSTB_CONTRACT_OVD	HASH	CONTRACT_REF_NO,EVENT_SEQ_NO, OVD_SEQ_NO
CSTB_EXT_CONTRACT_STAT	LIST	BRANCH_CODE
CSTB_MSG_LOG	HASH	MSG_ID
CSTB_RELATIONSHIP_LINKAGE	HASH	REF_NO
DETB_PCTRN	LIST	BRANCH
DETB_RTL_TELLER	HASH	TRN_REF_NO
ELTB_UTIL_TXN_LOG	HASH	MASTER_TXN_ID
FBTB_OVD	HASH	XREF,SEQ_NO
FBTB_TXNLOG_DETAILS	HASH	XREFID
FBTB_TXNLOG_MASTER	HASH	XREFID
FTTB_CONTRACT_MASTER	HASH	CONTRACT_REF_NO
GETB_MAIN_UTILS	HASH	UTIL_ID
GETB_UTILS	HASH	USER_REFNO
GETB_UTILS_LOG	LIST	UTIL_BRN
GETB_VD_UTILS	HASH	FACILITY_ID
GETH_UTILS	LIST	UTIL_BRN
GETM_LIAB	HASH	ID
GETM_LIAB_CUST	LIST	BRANCH_CODE
GLTB_CUST_ACCBREAKUP	LIST	BRANCH_CODE
GLTB_GL_BAL	LIST	BRANCH_CODE
GWTB_MSG_IN_LOG	HASH	MSG_REF_NO
GWTB_MSG_OUT_LOG	HASH	MSG_REF_NO
ICTB_ACC_PR	RANGE	BRN,PROCESS
ICTB_ACC_PR_HISTORY	LIST	BRN
ICTB_ADJ_INTEREST	LIST	BRN
ICTB_CHG_VAL	LIST	BRN
ICTB_DLY_MSG_OUT	LIST	BRN
ICTB_ENTRIES	RANGE	BRN,PROCESS
ICTB_ENTRIES_HISTORY	RANGE	BRN,PROCESS
ICTB_ICALC_STMT	LIST	BRN
ICTB_IS_VALS	LIST	BRN
ICTB_ITM_TOV	LIST	BRN
ICTB_UDEVAL_ROW	HASH	COND_KEY
ICTM_ACC	LIST	BRN
ICTM_CHILDTD_DETAILS	LIST	BRN
ICTW_ACC_PR	LIST	BRN

ICTW_MAKE_ROW	HASH	COND_KEY
ISTB_CONTRACTIS	HASH	CONTRACT_REF_NO
ISTB_CONTRACT_DETAILS	HASH	CONTRACT_REF_NO
ISTB_MSGHO	HASH	CONTRACT_REF_NO
MITB_CLASS_MAPPING	HASH	UNIT_REF_NO
MSTB_ADV_INPUT	HASH	DCN
MSTB_CONTRACT_CHG_ADVICE	HASH	CONTRACT_REF_NO
MSTB_EXT_MSG_OUT	HASH	DCN
MSTB_MSG_STAT	HASH	REFERENCE_NO
MSTM_MSG_ADDRESS	HASH	CUSTOMER_NO
SMTB_SMS_LOG	HASH	SEQUENCE_NO
STTB_ACCOUNT	LIST	BRANCH_CODE
STTB_FIELD_LOG	HASH	KEY_ID
STTB_NOTIFICATION	HASH	PKEY_VALUES
STTB_NOTIFICATION_HISTORY	HASH	PKEY_VALUES
STTB_RECORD_LOG	LIST	BRANCH_CODE
STTB_RECORD_MASTER	LIST	BRANCH_CODE
STTM_ACCSTAT_REPLINES_DETAIL	LIST	BRANCH_CODE
STTM_CUSTAC_BAL_NOTIF	LIST	BRANCH_CODE
STTM_CUSTOMER	HASH	CUSTOMER_NO
STTM_CUST_ACCOUNT	LIST	BRANCH_CODE
STTM_CUST_PERSONAL	HASH	CUSTOMER_NO
SWTB_TXN_HIST	HASH	XREF
SWTB_TXN_LOG	HASH	XREF
SWTB_TXN_TIME	HASH	XREF

Following points are to be noted during partitioning:

- Keep the number of partitions same as number of branches.
- All the corresponding indexes should be local partitioned except the primary key index which doesn't contain the partition key.
- If the primary key index doesn't contain the table partition column then those indexes are to be created without partition.
- 'Actb_daily_log and actb_history tables' primary key index has to be recreated as reverse key index without partition as follows:

```
Alter table ACTB_DAILY_LOG drop primary key;

Drop index PK01_ACTB_DAILY_LOG;

Create unique index PK01_ACTB_DAILY_LOG on ACTB_DAILY_LOG GLOBAL
PARTITION BY HASH (AC_ENTRY_SR_NO);

Alter table ACTB_DAILY_LOG add constraint PK01_ ACTB_DAILY_LOG
primary key (AC_ENTRY_SR_NO) using index PK01_ACTB_DAILY_LOG;
```

- Similarly recreate actb_history primary key.

- Index IX02_ACTB_DAILY_LOG on ACTB_DAILY_LOG (TRN_REF_NO,EVENT_SR_NO) should be recreated as global Hash partition as follows:

  ```
  Create index IX02_ACTB_DAILY_LOG on ACTB_DAILY_LOG (TRN_REF_NO,
  EVENT_SR_NO) global partition by hash (TRN_REF_NO, EVENT_SR_NO)
  partitions <no_of_partitions>;
  ```

- In ACTB_DAILY_LOG index, if there is an index on (AC_NO, AC_BRANCH, TRN_DT, TRN_CODE) columns, then this index can be dropped and create a new index as follows:

  ```
  CREATE INDEX X6_ACTB_DAILY_LOG ON ACTB_DAILY_LOG(AC_NO,AC_BRANCH
  ,BALANCE_UPD ,AUTH_STAT) local;
  ```

5. Sequence Caching

Sequence Caching is applicable only if Oracle Banking is deployed in RAC database.

Heavy use of sequences in RAC database causes high DFS lock handle & row cache lock waits which affect the application scalability. In order to overcome this issue, the sequences are to be cached with noorder option.

All the Banking indexes should be recreated cache 500 and noorder. Steps to alter existing sequences as follows:

1. Login to Banking schema
2. SQL > Spool sequence.sql
3. SQL > select ' alter sequence ' || sequence_name || ' cache 500 noorder;' from user_sequences;
4. SQL > spool off;
5. SQL > @ sequence.sql
   Verify that cache and order changed to all sequences.
6. Select order_flag, cache_size from user_sequences;

In Banking some of the sequences are recreated as part of end of day batches. Those sequences have to be taken care in TRPKS package. Sequence creation is handled in procedure Pr_Create_Seq and function Fn_Create_Seq_For_Combination. These methods should be modified to include caching and noorder as follows:

```
l_Create := 'CREATE SEQUENCE ' || p_Seq_Name || ' INCREMENT BY 1 START
WITH 1 MINVALUE 1 NOCYCLE CACHE 500 NOORDER';
```

6. PLSQL Optimizer Level

The plsql_optimize_level value for all the pl/sql units should be same which would be the value set in plsql_optimize_level init parameter.

Following sql gives the PLSQL optimizer level for Banking schema plsql units:

```
Select PLSQL_OPTIMIZE_LEVEL,type,count(*) "Count" from
user_plsql_object_settings group by PLSQL_OPTIMIZE_LEVEL,type;
```

PLSQL_OPTIMIZE_LEVEL for all the objects should be same which should be value set in plsql_optimize_level init parameter. If there is a difference then the objects should be recompiled. This can be done using dbms_utility.compile_schema procedure.

Eg: - exec dbms_utility.compile_schema('FCCBM2')

Here, ‘FCCBM2’ refers to the Banking schema.

Note: The ‘dbms_utility.compile_schema’ procedure invalidates and recompiles all the plsql units.

7. Statistics Collection for Banking Schema (Recommended Method)

Oracle 12c provides a default scheduled job to collect statistics for the entire database and is default scheduled to run every night. Given that the Banking batch as well runs in the night it is critical that the statistics gathering is not run during the batch.

It is recommended to use the default database scheduled job that is shipped with Oracle Database to collect statistics for Banking Schema.

Note: This document assumes that there is no other tool or a program is scheduled to collect statistics for the Database.

7.1 Customizing Default Statistics Collection Schedule

The Default Scheduler is to be customized for the following:

- Ensure that the default statistics gathering program is configured and Running.

  ```
  SELECT STATUS

  FROM DBA_AUTOTASK_CLIENT

  WHERE CLIENT_NAME='auto optimizer stats collection';
  ```

  Should return - ENABLED

- Ensure that the default statistics gathering program is configured to run only on weekends.

  ```
  /* Start of Script – Script to be executed as SYS*/

  BEGIN

  DBMS_AUTO_TASK_ADMIN.ENABLE(

      CLIENT_NAME => 'auto optimizer stats collection',

      OPERATION   => NULL,

      WINDOW_NAME => 'SATURDAY_WINDOW');

  DBMS_AUTO_TASK_ADMIN.ENABLE(

      CLIENT_NAME => 'auto optimizer stats collection',

      OPERATION   => NULL,

      WINDOW_NAME => 'SUNDAY_WINDOW');

  END;

  /
  ```

/* End of Script */

- Default schedule is daily. So disable the daily schedules for optimizer statistics.

```
/* Start of Script – Script to be executed as SYS*/
BEGIN
DBMS_AUTO_TASK_ADMIN.DISABLE(
    CLIENT_NAME => 'auto optimizer stats collection',
    OPERATION   => NULL,
    WINDOW_NAME => 'MONDAY_WINDOW');
DBMS_AUTO_TASK_ADMIN.DISABLE(
    CLIENT_NAME =>'auto optimizer stats collection',
    OPERATION   => NULL,
    WINDOW_NAME => 'TUESDAY_WINDOW');
DBMS_AUTO_TASK_ADMIN.DISABLE(
    CLIENT_NAME =>'auto optimizer stats collection',
    OPERATION   => NULL,
    WINDOW_NAME => 'WEDNESDAY_WINDOW');
DBMS_AUTO_TASK_ADMIN.DISABLE(
    CLIENT_NAME => 'auto optimizer stats collection',
    OPERATION   => NULL,
    WINDOW_NAME => 'THURSDAY_WINDOW');
DBMS_AUTO_TASK_ADMIN.DISABLE(
    CLIENT_NAME => 'auto optimizer stats collection',
    OPERATION   => NULL,
    WINDOW_NAME => 'FRIDAY_WINDOW');
END;
/
/* End of Script */
```

Verify the setup using the following SQL

```
SELECT WINDOW_NAME,OPTIMIZER_STATS

FROM DBA_AUTOTASK_WINDOW_CLIENTS;
```

Should return

```
MONDAY_WINDOW       DISABLED

TUESDAY_WINDOW      DISABLED

WEDNESDAY_WINDOW DISABLED

THURSDAY_WINDOW    DISABLED

FRIDAY_WINDOW         DISABLED

SATURDAY_WINDOW     ENABLED

SUNDAY_WINDOW       ENABLED
```

7.2 Customizing Statistics Gathering for Banking

The default statistics gathering is designed to be generic. It is recommended to customize the default statistics gathering to suit Banking online and batch.

Following are the areas that would need customization for Banking:

- Statistics Histograms
- Sample Size of Statistics

7.2.1 Statistics Histograms

Note the following:

- The default statistics gathering routine decides to collect histograms on specific tables based on certain criteria that are not documented.
- Statistics Histograms are not recommended for Banking tables.

Configure the default statistics gathered without Histograms.

```
/* Start of Script – Script to be executed as SYS*/

BEGIN

  DBMS_STATS.SET_PARAM ('METHOD_OPT','FOR ALL COLUMNS SIZE 1');

END;

/

/*End of Script */
```

Verify the setup using

```
SELECT DBMS_STATS.GET_PARAM ('METHOD_OPT') FROM DUAL;
```

Should return

```
FOR ALL COLUMNS SIZE 1
```

7.2.2 Sample Size of Statistics

The default statistics gathering routine decides on the percentage of data sampling (AUTO_SAMPLE_SIZE).

The idea of sampling is to reduce the time taken for collecting statistics. Sampling could be effective for very large historical tables but not for medium and small tables and hence Sampling of data for all Banking tables is not recommended

Configure the default statistics gathered with 100% data coverage.

```
/* Start of Script – Script to be executed as SYS*/

BEGIN

DBMS_STATS.SET_PARAM('ESTIMATE_PERCENT',100);

END;

/

/* End of Script */
```

Verify the setup using

```
SELECT DBMS_STATS.GET_PARAM('ESTIMATE_PERCENT')

FROM DUAL;
```

Should return

100

7.3 Script to Capture and Lock Stats for Volatile Tables in Banking Schema

As mentioned in section on Banking specific Statistic collection, statistics on the volatile tables are critical for performance and the statistics would have to be collected when these volatile tables have data.

The approach to be followed is as follows:

- Identify the time period where these specific tables have maximum data. E.g. ACTB_DAILY_LOG is an accounting table that is volatile. This table is bound to have maximum data (Peak Day of Business/ Month End Day).
- Unlock and Collect Statistics for this specific table on the day of Maximum Volume.
- Lock The statistics

Note: Different Banking tables might have different days of peak volume and hence the statistics should be collected at different days matching the peak volume for the respective table.

The statistics would have to be monthly refreshed so that the boundary values are refreshed. Lower bound and upper bound values are stored in the data dictionary and out dated boundary values might skew the cost of the SQL.

Use the attached script to capture statistics. The script would have to be run connecting as Banking schema. The following example uses ACTB_DAILY_LOG as the volatile table. The same script can be used for other tables as well.

```
Spool FCPAYMENTS_Vol_Table_Stats.txt

SELECT NUM_ROWS, BLOCKS, SAMPLE_SIZE, TO_CHAR(LAST_ANALYZED,'DD-
MON-YYYY HH24:MI:SS')

from USER_TAB_STATISTICS

WHERE TABLE_NAME='ACTB_DAILY_LOG';

exec dbms_stats.unlock_table_stats(USER,'ACTB_DAILY_LOG');

exec
dbms_stats.gather_table_stats(OWNNAME=>USER,tabname=>'ACTB_DAILY_
LOG',METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', CASCADE=>true,
DEGREE=>4);

exec dbms_stats.lock_table_stats(USER,'ACTB_DAILY_LOG');

SELECT NUM_ROWS, BLOCKS,SAMPLE_SIZE,TO_CHAR(LAST_ANALYZED,'DD-
MON-YYYY HH24:MI:SS')

from USER_TAB_STATISTICS

WHERE TABLE_NAME='ACTB_DAILY_LOG';

Spool off
```

8. Banking Database Storage Recommendations

Oracle database 10g release 2 onwards, Automatic Storage Management (ASM) is the recommended storage option for Banking database. ASM is an integrated cluster aware volume manager and a file system designed and optimized for managing Oracle database files. ASM is the realization of the Oracle Stripe and Mirror Everything (SAME) storage management methodology researched and established as best practices for Oracle database environment over many years.

Note: For configuring ASM refer Automatic storage management best practice document provided by Oracle for your database version.

8.1 Key benefits of ASM

- I/O is spread evenly across all available disk drives to prevent hot spots and maximize performance.
- ASM eliminates the need for over provisioning and maximizes storage resource utilization facilitating database consolidation.
- Inherent large file support.
- Performs automatic online redistribution after the incremental addition or removal of storage capacity.
- Maintains redundant copies of data to provide high availability, or leverage 3rd party RAID functionality.
- Supports Oracle Database 12c as well as Oracle Real Application Clusters (RAC).
- Capable of leveraging 3rd party multipathing technologies.
- For simplicity and easier migration to ASM, an Oracle Database 12c database can contain ASM and non-ASM files. Any new files can be created as ASM files whilst existing files can also be migrated to ASM.
- RMAN commands enable non-ASM managed files to be relocated to an ASM disk group.
- Oracle Database 12c Enterprise Manager can be used to manage ASM disk and file management activities.

9. Banking Database Backup Recommendations

Backup Policy is a very important ingredient of any High Availability system. Oracle recommends RMAN utility for database backup.

RMAN is acronym for Recovery Manager, is Oracle utility which will backup, restore, and recover oracle data files. RMAN is an Oracle provided utility for efficiently performing Backup and Recovery. RMAN is available as a part of the standard Installation and no separate installation is required.

Recovery Manager is a client/server application that uses database server sessions to perform backup and recovery. It stores metadata about its operations in the control file of the target database and, optionally, in a recovery catalog schema in an Oracle database.

You can invoke RMAN as a command-line executable from the operating system prompt or use some RMAN features through the Enterprise Manager GUI.

9.1 RMAN Vs Conventional Backup

- During a conventional hot backup, the amount of Redo generated during the backup would be more due to the fact that the redo logs during the hot backup store the entire block images rather than the change vectors.
- RMAN doesn't place the tablespace in a backup mode and hence the amount of Redo generated during the RMAN backup is considerably low.
- RMAN can identify block corruption during backup operations and RMAN supports Block recovery.
- RMAN automatically detects new data files and will backup them. Also, RMAN supports incremental backup method.
- RMAN backs up only the blocks that have been used at least once. Unused blocks are never backed up. Unused block here refers to the blocks where in the block header is zeroed
- RMAN enables us to test the backup without actually restoring the backup.
- RMAN can verify physical and logical structures of the database without actually performing backup.
- Usage of Shared Pool and Large Pool for RMAN
- RMAN uses DBMS_RCVMAN and DBMS_BACKUP_RESTORE packages for backup and recovery. These packages would be loaded in the shared pool for backup and restore operation. RMAN uses the PGA for backup and restore operation.
- RMAN Requires LARGE_POOL only if TAPE_IO_SLAVES and DBWR_IO_SLAVES are defined.
- **Sizing Large Pool** - LARGE_POOL = (Number of Channels) * (16 MB + Tape Buffer)

9.2 Benefits of Using RMAN

- RMAN is an intelligent tool that comes at no extra cost. It is available free with the Oracle Database.
- RMAN introduced in Oracle 8 it has become simpler with newer versions and easier than user managed backups.
- Provides proper security for Backups.
- You can be 100% sure your database has been backed up.
- Controlfile and Spfile of the database can be configured to be automatically backed up by RMAN.
- It contains detail of the backups taken etc in its central repository Facility for testing validity of backups also commands like crosscheck to check the status of backup.
- Faster backups and restores compared to backups without RMAN.
- RMAN is the only native backup tool which supports incremental backups.
- Oracle 12c has got further optimized incremental backup which has resulted in improvement of performance during backup and recovery time.
- Parallel operations (Multiple Channels for Backup and Restore)are supported.
- Better querying facility for knowing different details of backup.
- No extra redo is generated when backup is performed, compared to conventional online backup.
- Maintains repository of backup metadata.
- Remembers backup set location.
- Knows what need to backed up.
- Knows what is required for recovery.
- Knows what backup are redundant.
- RMAN can back up the Database to Disk or directly to Tape. It is recommended that RMAN backup is performed to disk and then copied to tape.

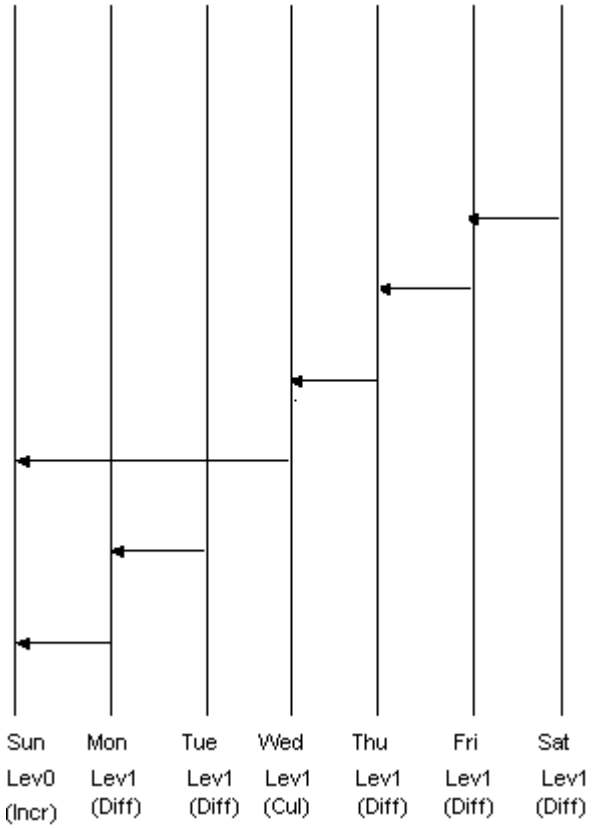
9.3 Backup Strategy Recommendation

RMAN will not backup the below files so it is advised to take the copy of the below files on regular basis (weekly/any change/addition to the file).

- Tnsnames.ora
- Listener.ora
- Password file
- Init.ora

The Best practice is to take create the pfile once the spfile is updated.

Below is the recommended strategy.



10. Banking Periodic Table Maintenance

Find below the list of tables and the corresponding action that needs to be planned in a periodical basis so that no performance degradation is observed over a period of time. These activities need to be planned ahead since this would require downtime. Also ensure proper backups are taken prior to any table maintenance activity.

Details of the various Actions are as below:

Truncate Table: Take the backup of the current table data and truncate the table

Compress Table: Data of this table is required hence compress table and index data

Recreate Table: These tables and it corresponding indexes needs to be rebuild.

Table Name	Action
FBTB_TXNLOG_DETAILS_HIST	Truncate Table
CSTB_MSG_LOG	Truncate Table
SMTB_IMAGE_UPLOAD	Truncate Table
FBTB_TXNLOG_DETAILS_HIST	Compress Table
ACTB_ACCBAL_HISTORY	Compress Table
ACTB_DAILY_LOG	Recreate table
MITB_CLASS_MAPPING	Compress Table
ACTB_HISTORY	Compress Table
SVTM_UPLOAD_CIF_SIG_DET	Truncate Table
FBTB_TXNLOG_DETAILS_HIST	Truncate Table
SVTM_CIF_SIG_DET	Compress Table
CSTB_MSG_LOG	Truncate Table
STTB_FIELD_LOG	Recreate table
STTM_CUST_IMAGE	Compress Table
ICTB_ICALC_STMT	Recreate table
STTB_FIELD_LOG_HIST	Truncate Table
FBTB_TXNLOG_MASTER_HIST	Truncate Table

STTB_FIELD_LOG_HIST	Compress Table
STTB_FIELD_LOG	Recreate table
DETB_RTL_TELLER	Recreate table
STTB_RECORD_LOG	Recreate table
SWTB_TXN_HIST	Recreate table
ICTB_ENTRIES_HISTORY	Compress Table

Note: For Maintenance Activity related to Truncate and Recreate table, the impact to be analyzed at site level before implementing the action. Also for any purging related solutions required, please refer the document FS_FCPAYMENTS_12.0.2_CO_Purging.docx .

11. Banking Functionality Related Performance Changes

The following parameters are discussed on functionality related performance changes:

Parameter	Recommended Value	How to find
Real debug parameter	N	select param_val from cstb_param where param_name='REAL_DEBUG'
ONLINE GL Update	N	select ONLINE_GL_UPDATE from STTM_BANK
VD Balance update	OFFLINE	select param_val from cstb_param where param_name='VDBAL_UPDATE'
CL - Netting - Accrual	Y	select GL_NETTING_ACCR from CLTM_BRANCH_PARAMETERS
CL - Netting - Liquidation	Y	select GL_NETTING_LIQD from CLTM_BRANCH_PARAMETERS
CL - Netting - STCH	Y	select GL_NETTING_STCH from CLTM_BRANCH_PARAMETERS

12. Appendix

12.1 Script to Check Histograms on Banking Schema

Following script would have to be executed in the Banking schema:

```
select distinct table_name

from

(

select table_name from user_tab_columns where histogram!='NONE'

)
```

Should return **No Records**

12.2 Script to Remove Histograms on Banking Schema

Following script would have to be executed in the Banking schema if there are any rows:

```
declare

cursor cur_tables is

select distinct table_name

from

(

select table_name from user_tab_columns where histogram!='NONE'

);

begin

for rec_tables in cur_tables

loop

dbms_stats.gather_table_stats(ownname=>USER,tabname=>rec_tables.t
able_name,METHOD_OPT=>'FOR ALL COLUMNS SIZE
1',CASCADE=>TRUE,DEGREE=>2,ESTIMATE_PERCENT=>NULL);

end loop;

end;
```

Oracle Banking Payments
Database Practices DB 12c
[November] [2018]
Version 14.2.0.0.0

Oracle Financial Services Software Limited
Oracle Park
Off Western Express Highway
Goregaon (East)
Mumbai, Maharashtra 400 063
India

Worldwide Inquiries:
Phone: +91 22 6718 3000
Fax:+91 22 6718 3001
www.oracle.com/financialservices/