Using Perceptive Content with an Oracle Database

Best Practices

Version: Foundation EP4

Written by: Product Knowledge, R&D
Date: June 2021

Hyland™

Copyright

Table of Contents

About the Oracle database server

Perceptive Content works with several different databases, including Oracle. Oracle presents unique challenges due to the vast configuration possibilities that exist. This document provides basic guidelines for configuring your Oracle database based on industry-wide best practices, testing, and client feedback.

Reference this guide while you are creating, configuring, or tuning your Oracle database. For additional information about creating your database, please refer to the *Perceptive Content Installation and Setup Guide*.

You can also refer to the Quick Reference Guide at the end of this document, which provides most of this information in a checklist format.

Getting started

The following sections contain information you should consider when preparing for an Oracle database.

These components are the backbone of your infrastructure and play a critical role in the performance, availability, scalability, security and longevity of your database.

- Oracle edition
- Host configuration
- Storage configuration
- Network configuration
- Version and patching
- Administration

Oracle edition

You can use Perceptive Content with Oracle Standard Edition or Enterprise Edition. Be sure to review the features available with each edition before making a choice. For larger customers, we recommend using Enterprise Edition due to the inclusion of features associated with High Availability (Data Guard and Online operations), Scalability (RAC), and Manageability (Diagnostic and Tuning Packs). These are all important features to have for business critical database environments.

Please review the *Perceptive Content Technical Specifications* documentation for a list of supported Oracle database versions for the version of Perceptive Content you are deploying.

Host configuration

A number of different operating systems can host the Oracle database including Oracle Linux, AIX, HP/UX, Solaris and Windows. To ensure adequate support for your environments, use a platform that your technical staff can most effectively support based on their experience as well as factoring in operating costs and performance.

Ensure the host has a sufficient number of sockets/processors and physical memory to accommodate the anticipated demands based on the number of users and size of the database. Proper capacity planning should take place to ensure a host is provisioned that provides adequate resources to deliver a highly performant environment.

Storage

There are a number of different storage providers available to use for database storage. You should base your choice of storage on several factors including performance, cost, supportability, scalability, compatibility, management, and longevity. Ensure that your storage option can accommodate the needs for long term growth and availability while maintaining high performance for both read and write operations.

Network configuration

Configure an isolated and secure network topology for your databases. Configure the network environment used for communication between the database and the application servers to use encryption and secure protocols. Use DNS to accommodate your implementation of Oracle with consideration to GRID (Clusterware/ASM) configurations, including SCAN addresses. We recommend using SSL/TCPS for all network traffic to the database, including SCAN listeners, local listeners, and private cluster interconnect traffic.

Version and patching

Refer to the *Perceptive Content Technical Specifications* for the required database versions that are supported for each version of Perceptive Content.

We recommend you patch your Oracle environments regularly to ensure any security vulnerabilities are minimized and to ensure that bug fixes are applied when available. This applies to the host, network appliances, storage appliances and the database.

Be sure to thoroughly validate any new patches in non-production environments before promoting them to production to help minimize issues associated with performance regression or instability.

Administration

Having skilled administrators to manage and maintain your infrastructure is very important. There are many components that have to work seamlessly to deliever a highly performing and secure environment. The importance of having individuals that can effectively configure, tune, and troubleshoot the various components of your infrastructure cannot be understated. This applies to databases, storage, networking, system administration, and management.

Set up

During the creation or configuration of your Oracle database, perform the tasks in the following sections.

- Oracle instance
- Storage
- Tablespaces
- Datafiles
- Undo/Rollback segment management
- Control files
- Redo logs
- Fast Recovery Area

- Archivelog mode
- System Global Area (SGA)
- Processes
- Character sets
- Connection mode
- Oracle parameters
- Change default passwords

Oracle instance

Every Oracle database is associated with an Oracle instance. When you create a database on the server, Oracle allocates an area in memory called the System Global Area (SGA) and starts one or more Oracle processes. This combination of the SGA and the Oracle processes comprises an Oracle instance. The memory and processes of an instance manage the associated database's data and serve the users of the database.

While not required, we recommend that the Oracle instance or database hosting the Perceptive Content database schema be dedicated to serving only the Perceptive Content database. This dedication provides you with the flexibility to make specific configuration changes as necessary for the Perceptive Content instance or database without having to factor in other environments.

Storage

Oracle recommends using Oracle Automatic Storage Management (ASM) for database storage management. Refer to Oracle documentation for complete details as you plan and configure your storage for your Oracle databases.

If using a file system (not ASM) for database storage, be sure to choose a file system on physical devices that are dedicated to the database.

Oracle recommends the SAME approach (Stripe and Mirror Everywhere). Raid 1+0 is recommended only for the database files. Raid 1 is recommended for the others.

The following table provides Oracle's recommended RAID use with each type of Oracle database file.

RAID	Type of RAID	Control File	Database File	Redo Log File	Archive Log File
0	Striping	Avoid	OK	Avoid	Avoid
1	Shadowing	Best	OK	Best	Best
1+0	Striping and Shadowing	OK	Best	Avoid	Avoid
3	Striping with static parity	OK	OK	Avoid	Avoid
5	Striping with rotating parity	OK	Best if RAID 0-1 are not available	Avoid	Avoid

Tablespaces

The INUSER schema has two possible configurations with respect to tablespaces and the location of each table and index within the tablespaces. The basic, standard configuration only consists of two tablespaces called DATA (for tables) and INDX (for indexes). The Advanced Tablespace configuration contains 116 different tablespaces, 58 for tables and 58 for indexes. Tables and indexes are strategically located across the different tablespaces to help facilitate the distribution of IO across multiple, physical devices.

You should modify the database schema creation scripts, SunflowerORA.sql or SunflowerORA-AdvancedTS.sql, before execution to define the paths to where the system creates the datafiles.

Your database administrator can pre-create the tablespaces or can modify the creation script as needed to utilize Oracle-Managed Files (OMF).

You can move individual tables or indexes into different tablespaces as necessary for general maintenance, to help distribute the I/O between database files, or to make use of different block sizes for performance.

Datafiles

You can designate tablespaces as either BIGFILE or SMALLFILE tablespaces. We recommend using SMALLFILE tablespaces to provide you with the flexibility to create multiple datafiles for a single tablespace. BIGFILE tablespaces can contain only one file, whereas traditional SMALLFILE tablespaces can contain up to 1,022 files. We recommend that all data and index tablespaces be of the smallfile type.

Undo/Rollback segment management

We recommend that you use the following undo management settings and parameters as a starting point. You should periodically review the Undo Advisor recommendations to determine an appropriate value for the undo_retention parameter for your environment.

Name	Value
undo_management	AUTO
undo_retention	9000
undo_tablespace	<Undo Tablespace Name>

As you increase the undo_retention parameter, you must also increase the size of the undo tablespace to allow for the higher retention. You may need to adjust these parameters appropriately in the event you encounter ORA-1555 errors or other errors related to undo retention.

Control files

We recommend that you multiplex the control files for your database to provide greater fault tolerance.

Ensure your database has at least two control files located on separate disks. If a control file fails, you can restore it using the intact copy of the control file from the other disk. Specify the disk locations in the database's initialization parameter file. Control files store the status of the database physical structure, which is crucial to database operation. Make sure you back up your control files during your scheduled backups and after any configuration changes such as changes to control files, redo log, or datafiles.

Redo logs

We recommend that you multiplex the redo log files for your database to provide greater fault tolerance.

We recommend that you maintain at least three redo log groups with two members per group. Make sure each member of a redo log group is located on different physical disks for effective multiplexing. Create all redo logs with the same size.

The number and size of your redo logs depends on your volume. If you have a very high rate of activity (inserts, updates, and deletes), consider increasing the size of redo log files to prevent them from switching too often. If necessary, you may also want to create additional redo log groups to allow the database more time to write the changes to disk and help prevent from having to wait for log switches. Similarly, do not create unnecessarily large redo logs because they may not switch often enough and can lead to increased data loss in the event of media failure. If your database unexpectedly shuts down or you lose all members of an active redo log group, large redo logs increase the potential for higher data loss. We recommend that you create your redo logs to approximately 256MB or 512MB so that the log naturally switches about every 15 minutes during peak hours. You may also decide to write custom scripts to force a log switch at specific intervals to help prevent against data loss.

Fast Recovery Area

We recommend configuring the Fast Recovery Area and allocating a sufficient amount of storage to the fast recovery area disk group on storage devices that can contain at least three days of recovery information. Refer to Oracle documentation for details on configuration and calculating the appropriate disk space to allocate for your particular environments.

Archivelog mode

For production environments, we recommend running in archivelog mode to facilitate inconsistent backups which can be used for media and point in time recovery. If you are running Oracle in archivelog mode, verify the archive destination is large enough to contain at least five days' worth of normal archivelog data. Maintaining this space helps to ensure the destination does not fill up and cause the database to hang. Archived logs must be backed up before being removed to ensure recoverability.

System Global Area

The System Global Area (SGA) is a group of shared memory structures that contains data and control information for the Oracle database. When an Oracle database instance starts, it allocates an SGA.

Ensure your host is configured with enough physical memory to accommodate the anticipated SGA size based on future capacity planning.

We recommend that you enable Automatic Shared Memory Management (ASMM) to let Oracle determine the best way to allocate the SGA between each component. The following chart represents a starting point for your Oracle SGA.

Value	Non-Dedicated	Dedicated
SGA	10GB	70% of total RAM
PGA	2GB	

If you have available memory, consider allocating additional memory by setting the Maximum SGA Size parameter. The Maximum SGA Size specifies the maximum memory that the database can allocate. Specifying the Maximum SGA Size adds flexibility to dynamically increase the total SGA size up to the maximum SGA size if needed.

Because the SGA stores data in memory for fast access, the SGA should be within main memory. If you swap pages of the SGA to disk, then the data is no longer quickly accessible. On most operating systems, the disadvantage of paging significantly outweighs the advantage of a large SGA.

Paging occurs when an operating system transfers memory-resident pages to disk solely to allow new pages to be loaded into memory. Many operating systems page to accommodate large amounts of information that do not fit into real memory. On most operating systems, paging reduces performance.

Use your operating system utilities to examine the operating system, to monitor paging on your system. If there is significant paging, then the total memory on the system might not be large enough to hold everything for which you have allocated memory. Either increase the total memory on your system, or decrease the amount of memory allocated.

For operating systems that support HugePages then consider configuring a sufficient number of HugePages so that the entire SGA can be allocated using HugePages.

For operating systems that support pre-paging and locking of the SGA, consider setting the pre_page_sga and lock_sga parameters to true.

As the database grows you will need to adjust the values for the SGA, and other memory components, accordingly based on your environment. We recommend that you periodically evaluate the Oracle buffer pool statistics as well as the advisory statistics for the various memory components to ensure they are set to optimal values.

Processes

You should set the processes parameter to a value that accommodates all of the Oracle background processes as well as the Perceptive Content application server pool connections and any additional connections such as external interfaces or third party monitoring agents.

Normally, a single instance of the Perceptive Content application server establishes about 155 database connections. For an active-active configuration of the Perceptive Content application server, you should double the expected number of database connections attributed to the application servers.

Be sure to review the processes parameter to ensure it can accommodate the total number of expected connections any time the number of database connections is expected to increase.

Character sets

Perceptive Content does not currently support a Unicode configuration on Oracle. therefore, the default Unicode character set of AL32UTF8 is not supported. To ensure compatibility between the server and the database, and to reduce the likelihood of non-supported characters being inserted into the database, please use the WE8MSWIN1252 character set.

Perceptive Content does not use Unicode character types (NVARCHAR2, NCHAR) and should use the default National Character Set of AL16UTF16.

Connection mode

The Perceptive Content server establishes and maintains its own database connections pools. The connections remains open and idle until needed by the server. As a result, configuring your Oracle database to use the Dedicated server mode is preferred over using the Shared server mode.

Oracle parameters

Session parameters

The following parameters ensure that Perceptive Content performs as intended and helps prevent configuration issues. Each connection that the application makes to the database sets these parameters by altering the session after the connection is established.

These parameters should also be used for manual query execution and for any performance tuning tasks to ensure the queries are optimized by facilitating the usage of the function-based indexes.

```
NLS_DATE_FORMAT = 'RRRR-MM-DD'
NLS_TIME_FORMAT = 'HH24:MI:SS.FF'
NLS_TIMESTAMP_FORMAT = 'RRRR-MM-DD HH24:MI:SS.FF'
NLS_COMP=LINGUISTIC
NLS_SORT=BINARY_CI
OPTIMIZER_MODE = ALL_ROWS
QUERY_REWRITE_INTEGRITY = TRUSTED
QUERY_REWRITE_ENABLED = TRUE
CURSOR_SHARING = EXACT
OPTIMIZER_INDEX_COST_ADJ=1
```

Inow.ini [ODBC] parameters

The following additional session level parameters can be defined on the application server if needed for performance tuning. These parameters are set according to the values in their respective inow.ini odbc parameters. Please reference the *Perceptive Content Update Guide* for more details on the default values and options for these parameters.

```
_replace_virtual_columns
optimizer_cost_based_transformation
optimizer_dynamic_sampling
optimizer_index_cost_adj
```

Database parameters

The following system parameters are recommended as a result of internal testing or customer feedback or required based on functionality of the product.

PRE_PAGE_SGA

PRE_PAGE_SGA determines whether Oracle reads the entire SGA into memory at instance startup. Operating system page table entries are then prebuilt for each page of the SGA. This setting can increase the amount of time necessary for instance startup, but it is likely to decrease the amount of time necessary for Oracle to reach its full performance capacity after startup. We recommend setting PRE_PAGE_SGA to TRUE.

LOCK_SGA

LOCK_SGA locks the entire SGA into physical memory. It is usually advisable to lock the SGA into real (physical) memory, especially if the use of virtual memory would include storing some of the SGA using disk space. For operating systems that support this, we recommend setting LOCK_SGA = TRUE.

MAX_IDLE_TIME

The MAX_IDLE_TIME parameter specifies the maximum number of minutes that a session can be idle before it is automatically terminated. The default value for this parameter is 0, which indicates that there is no limit. The default setting of 0/unlimited is required for the Perceptive Content database and is the only supported value for this parameter.

The premature termination of database connections within a pool can lead to instability within the application. To keep the database connection pools in a healthy state, you must not set maximum connection or maximum idle time limitations for the Perceptive Content database connections by using the maximum_idle_time parameter or through the use of Oracle Profiles or the Oracle Resource Manager plan directives.

Change default passwords

Change the default passwords for all the user accounts that are created during installation, including SYS, SYSTEM, SYSMAN, DBSNMP, and OUTLN, to help prevent against intrusion. When creating passwords, use complex passwords and also change the default passwords for the Perceptive Content database users, INUSER and INEMUSER.

Maintenance

The following sections provide basic guidance on how to maintain your database after creating it.

- Backups
- Statistics
- Automatic Database Diagnostics Monitor (ADDM) reporting
- Index rebuilds

Backups

Backing up your database regularly is one of the most important tasks of database administration.

For maximum recoverability we recommend that you configure your database to run in archivelog mode and schedule daily online backups of the database and frequent archive log backups using RMAN to disk or tape. If you backup to disk, you should have third party media management software to transfer your RMAN backups to tape.

At a minimum you should perform daily data pump exports of your database.

As with any backup solution, perform periodic tests to ensure you can restore and recover your database to a predefined point in time. This process helps you prepare for recovery from various situations.

Statistics

There are two types of automatic statistics collected in Oracle by default. Statistics stored in the Automatic Workload Repository (AWR) and statistics collected for the Cost-Based Optimizer (CBO). The following sections provide a high-level explanation of each type of statistic and provide some basic recommendations and examples.

Automatic Workload Repository (AWR)

AWR consists of operational statistics serving diagnostic, self-tuning and database tuning purposes. AWR statistics reflect the state of the whole database and various areas within the database during specific periods. It includes information about wait events, latches, enqueues, CPU consumption, SGA components, PGA, various statistics and metrics, most expensive SQLs by different measures, and so on. AWR automatically collects these statistics and you can access them in AWR reports.

By default, the AWR takes a snapshot of performance related statistics every 60 minutes and retains the statistics for seven days. The default and recommended Collection Level is "TYPICAL".

Modifying AWR snapshot settings

You can adjust the interval, retention, and captured Top SQL of snapshots generated for a specified database ID, but these adjustments can affect the precision of the Oracle diagnostic tools. The INTERVAL setting affects how often in minutes that snapshots are automatically generated. The RETENTION setting affects how long in minutes that snapshots are stored in the workload repository.

The TOPNSQL setting affects the number of Top SQL to flush for each SQL criteria (Elapsed Time, CPU Time, Parse Calls, Shareable Memory, and Version Count). The value for this setting is not affected by the statistics or flush level and overrides the system default behavior for the AWR SQL collection. It is possible to set this setting to MAXIMUM to capture the complete set of SQL in the cursor cache. However, setting it to MAXIMUM or a high value may lead to possible space and performance issues since there will be more data to collect and store. To adjust the settings, use the MODIFY_SNAPSHOT_SETTINGS procedure as shown in the following example:

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS
 (
  RETENTION => 43200,
  INTERVAL  => 60,
  TOPNSQL   => 100,
  DBID      => 4249447005
 );
END;
/
```

Note STATSPACK also provides this type of statistic but it does not store statistical data in the AWR structures. You should not run STATSPACK if you are gathering AWR statistical data due to the potential of performance degradation when both are running. If necessary, you can uninstall Statspack by running the spdrop.sql script.

Baseline AWR reports

Once an environment is in a state that performs well you should gather baseline AWR reports for an entire week as well as for each day of the week and maybe even for certain high volume periods. Having those baseline reports would be helpful to compare against down the road if you start experiencing a degradation in performance. Also, by default, the AWR data only has an 8-day retention period. We recommend that you increase it to 30 days.

Optimizer statistics (for Cost Based Optimizer)

The CBO maintains statistics about various database objects, such as, tables and indexes, in both the application schemas and Oracle internal schemas (SYS and SYSTEM). These statistics include the number of rows in each table, number of distinct values in columns (densities), average length of columns, and distribution of values within a column (histograms).

The Oracle Cost-Based optimizer uses these statistics for calculating the optimal execution plans for a query. The query along with the configuration and session parameters constitute the input for the optimizer to determine the best execution plan for each SQL statement.

The goal when gathering statistics is to generate them with as much statistical accuracy as possible, therefore higher sample sizes are suggested because any reduction in sample sizes can affect the accuracy of the statistics. Note that 100% sample sizes are potentially time consuming so you may need to plan carefully to fit the statistics gathering activities within the existing maintenance windows.

Normally, gathering new optimizer statistics should maintain or improve existing execution plans, but occasionally the performance of certain queries may degrade if updated stats result in suboptimal query plans.

Automatic Statistics Gathering tasks

The default GATHER_STATS_JOB jobs are responsible for collecting optimizer statistics. Oracle automatically collects statistics for all database objects, which are missing statistics or have stale statistics (data has changed by 10 percent) during predefined maintenance windows (10pm to 2am weekdays and 6am to 2am at the weekends).

You can change the maintenance window schedules using Enterprise Manager or the DBMS_SCHEDULER and DBMS_AUTO_TASK_ADMIN packages. You can also change the percentage for staleness as well as other default values using the SET_DATABASE_PREFS procedure, SET_GLOBAL_PREFS procedure, SET_SCHEMA_PREFS procedure and SET_TABLE_PREFS procedure.

Other options include gathering extended statistics, saving statistics, locking statistics as well as collecting different versions of statistics, using different methods, and switching between them to determine which method produces the best performance.

You can use the following command to disable the default statistics job if you are planning to use custom scripts:

```
Connect as SYS
EXEC DBMS_SCHEDULER.DISABLE('GATHER_STATS_JOB');
```

If the default, automatic statistics job is disabled for any reason then be sure to use other user defined DBMS_STATS scripts instead.

Manual statistics gathering

Unless the auto gathered stats are producing sub-optimal plans then there shouldn't be a need to manually collect statistics or alter how stats are gathered.

Guidelines for setting the sample size

In the context of optimizer statistics, sampling is the gathering of statistics from a random subset of table rows. By enabling the database to avoid full table scans and sorts of entire tables, sampling minimizes the resources necessary to gather statistics.

Oracle recommends using the AUTO_SAMPLE_SIZE value within the DBMS_STATS package for updating statistics. This is the default behavior and should be used in most cases.

There are times when it might be necessary to gather statistics by specifying a specific sample size (estimate_percent) including 100% or compute (NULL) or deleting histograms but those cases should be outliers to the norm.

ESTIMATE_PERCENT

The ESTIMATE_PERCENT parameter determines the percentage of rows used to calculate the statistics. The most accurate statistics are gathered when all rows in the table are processed (for example, a 100% sample), often referred to as computed statistics.

Starting with Oracle Database 11g, a new sampling algorithm was introduced that is hash based and provides deterministic statistics. This new approach has an accuracy close to a 100% sample but with the cost of, at most, a 10% sample. The new algorithm is used when ESTIMATE_PERCENT is set to AUTO_SAMPLE_SIZE (the default) in any of the DBMS_STATS.GATHER_*_STATS procedures.

It is highly recommended that from Oracle Database 11g onwards that the default AUTO_SAMPLE_SIZE is used for ESTIMATE_PRECENT. This is especially important on Oracle 12c because of the new histogram types, HYBRID and Top-Frequency, which can only be created if an auto sample size is used.

METHOD_OPT

The METHOD_OPT parameter controls the creation of histograms during statistics collection. Histograms are a special type of column statistic created to provide more detailed information on the data distribution in a table column. The default and recommended value for METHOD_OPT is 'FOR ALL COLUMNS SIZE AUTO', which means that histograms will be created for columns that are likely to benefit from having them.

A column is a candidate for a histogram if it is used in equality or range predicates such as WHERE column = 'X' or WHERE column BETWEEN 'A' and 'B' and, in particular, if it has a skew in the distribution of column values.

The optimizer knows which columns are used in query predicates because this information is tracked and stored in the dictionary table SYS.COL_USAGE$.

Recommended approach for manual statistic collection

You should gather statistics for the database objects at periodic intervals (daily, weekly or monthly) depending on how fast your data changes.

Note It may be necessary to make adjustments down the road as necessary for specific tables.

First Time/Weekly statistics

Run the First Time/Weekly version, seen below, during the weekend maintenance window. Ensure that it fits in the maintenance windows. This is the Oracle default.

Oracle - Gather Statistics for the entire INUSER schema

```
BEGIN
 DBMS_STATS.GATHER_SCHEMA_STATS
 (
  OWNNAME          => 'INUSER'
 ,ESTIMATE_PERCENT => DBMS_STATS.AUTO_SAMPLE_SIZE
 ,METHOD_OPT       => 'FOR ALL COLUMNS SIZE AUTO'
 ,CASCADE           => TRUE
 ,DEGREE           => DBMS_STATS.AUTO_DEGREE
 ,OPTIONS          => 'GATHER'
 );
END;
/
```

Nightly statistics

Run the Nightly version so that any objects that do not have statistics or that might reach the threshold of stale are updated that night.

Note OPTIONS => GATHER_AUTO only gathers necessary statistics. Oracle implicitly determines which objects need new statistics and determines how to gather those statistics.

Oracle - Gather Missing or Stale Statistics for the entire INUSER schema

```
BEGIN
 DBMS_STATS.GATHER_SCHEMA_STATS
 (
  OWNNAME          => 'INUSER'
 ,ESTIMATE_PERCENT => DBMS_STATS.AUTO_SAMPLE_SIZE
 ,METHOD_OPT       => 'FOR ALL COLUMNS SIZE AUTO'
 ,CASCADE           => TRUE
 ,DEGREE           => DBMS_STATS.AUTO_DEGREE
 ,OPTIONS          => 'GATHER AUTO'
 );
END;
/
```

Table level statistics

You can update statistics as needed for specific tables during troubleshooting or after abnormal operations that might render previously collected statistics inadequate.

Gather statistics for the specified table and its indexes

```
BEGIN
 DBMS_STATS.GATHER_TABLE_STATS
 (
  OWNNAME => 'INUSER'
 ,TABNAME => 'IN_INSTANCE'
 ,ESTIMATE_PERCENT => DBMS_STATS.AUTO_SAMPLE_SIZE
 ,METHOD_OPT => 'FOR ALL COLUMNS SIZE AUTO'
 ,CASCADE => TRUE
 ,DEGREE => DBMS_STATS.AUTO_DEGREE
 );
END;
/
```

Consideration for Weekly vs. Nightly statistics

Consider the following questions when trying to determine which statistics to gather. How much does the data change? How big is the database? How long does the First time/Weekly version take to run and does it comfortably fit into the maintenance windows?

The idea of the Nightly version (GATHER_AUTO) is to only update stats for any objects that do not have stats or have statistics that are considered stale due to the data having changed by a certain percentage (due to inserts/updates/deletes). By default, an object is considered stale once 10% of the data has changed.

If performance is always better on Mondays (maybe due to having better stats after the weekly job) and the weekly version comfortably fits into the nightly maintenance window then you could also run the default weekly version (OPTION=>GATHER) nightly instead of GATHER_AUTO. That would be the easiest approach and requires less effort than manually adjusting parameters for individual tables. The

larger the database and the duration of these jobs the more likely you will want to consider a more granular approach to updating stats.

Considerations for adjusting the STALE_PERCENT for specific tables

It is possible that you might want to reduce the stale_percent for some of the larger tables so that the daily job picks them up more frequently during the week. You can manually change and reduce the stale percentage for any tables. For example, if you see better day-to-day performance from updating stats after 5% causing the system to update the statistics more frequently for those tables. In contrast, you may also find it beneficial to increase the stale percentage for certain tables to prevent the system from updating statistics too often. In general, it is not recommended to make changes to the stale percentage.

Considerations for locking statistics

In some cases you may find that a certain set of statistics result in optimal performance for specific tables and result in the best query plans. In those cases, you may choose to lock statistics for specific tables to help promote plan stability. In general, it is not recommended to lock statistics for most tables unless thorough testing and performance evaluation has been conducted.

System statistics

System statistics allow the optimizer to consider a system's I/O and CPU performance and utilization. For each candidate plan, the CBO optimizer computes estimates for I/O and CPU costs. It is important to know the system characteristics to pick the most efficient plan with the optimal proportion between I/O and CPU cost. System CPU and I/O characteristics depend on many factors and do not stay constant all the time. The gathered statistics are:

- Single block readtime in ms.
- Multiblock readtime in ms.
- CPU speed in mhz.
- Average multiblock_read_count in number of blocks.

System statistics help the optimizer make decisions based on cost in proportion to elapsed time. This provides the optimizer with information to better judge if a full table scan or an index access is more appropriate. You should only need to gather system statistics initially and then again after any significant changes to the system. Statistics you gather initially and after system changes reflect the baseline. You should also capture statistics during an interval of time when the system has the most common workload.

Unlike when table, index, or column statistics are updated, the Oracle server does not invalidate already-parsed SQL statements when system statistics get updated. Only the new parsed SQL statements use the newly activated statistics. Use the following example to gather system statistics.

```
execute dbms_stats.gather_system_stats('Start');
```

After an adequate amount of time based on workload, use the following example to stop gathering system statistics.

```
execute dbms_stats.gather_system_stats('Stop');
```

To see the statistics, query a table (owned by the SYS schema) called aux_stats$. After gathering system statistics, this table contains a few critical numbers used by the new optimizer algorithms to calculate costs.

The exact list of results, using the query below, depends on your version of Oracle, but the following example provides typical results:

The following query provides the current settings.

```
Select pname, pval1
from sys.aux_stats$
where sname = 'SYSSTATS_MAIN';

PNAME                PVAL1
-------------------------
CPUSPEEDNW      2121.72855
IOSEEKTIM               10
IOTFRSPEED            4096
SREADTIM             2.235
MREADTIM             4.339
CPUSPEED              2134
MBRC                    13
MAXTHR              8578048
SLAVETHR
```

Automatic Database Diagnostics Monitor (ADDM) reporting

Along with the AWR reports you can also use the ADDM reports to analyze a snapshot range. These reports are a good place to start for getting tips to tune your database. Run the ADDM report (addmrpt.sql) from within OEM or call the report directly from the $ORACLE_HOME/rdbms/admin directory.

Index rebuilds

Index rebuilds are not normally necessary unless an index is highly fragmented and data access often requires sequential multi-block reads that are negatively impacted by fragmentation. Indexes can become fragmented over time due to inserts, updates, and deletes (DML). You can run the following command to check if an index is a good candidate for re-indexing.

Note Running the analyze command with the validate structure clause puts an exclusive lock on the table so run this command during off-peak hours to minimize the impact to users.

```
analyze index inuser.<Index_Name> validate structure;

select name as IndexName,
       height,
       lf_rows,
       del_lf_rows,
       DEL_LF_ROWS_LEN,
       USED_SPACE,
       PCT_USED
   from index_stats;
```

After you run the command and gather the data, we recommend rebuilding an index if it meets the following criteria.

- Index levels (height) is greater than 3
- PCT_USED is less than 75%.
- Rows deleted are greater than 20% (space is not automatically reused).
- Index is unclustered and performance is degrading (causing increases in number of blocks to be read).

Tuning

The following sections include additional tuning options that can be considered for specific tables and indexes if needed.

Note Adjusting the INITRANS and PCTFREE parameters, as a part of tuning, is not generally necessary but can be considered if standard tuning efforts with statistics and indexes are not enough.

INITRANS

By default, Oracle sets INITRANS to 1 for tables and 2 for indexes. In some cases, you may benefit by slightly increasing this parameter to a higher number. If you are seeing excessive ITL waits associated with a specific table or index then consider setting INITRANS to 5 or so then re-evaluate the effcts. The maximum setting for this parameter is 255, assuming there is enough free space in the block to accommodate the transaction. Each transaction uses 24 bytes. You can modify a table and optionally rebuild it to increase INITRANS.

PCTFREE

The default value for PCTFREE is 10, which results in reserving 10% of each block for future updates. You can increase the PCTFREE parameter on certain tables to help improve performance in various ways. For example, you can increase PCTFREE in order to reserve a percentage of each block so that rows can grow as needed due to updates, so that row chaining is less likely to occur. Increasing the amount of free space also allows for more ITL's (Interested Transaction List) slots in each block. Increasing PCTFREE along with smaller block sizes can further reduce the number of rows stored in a single block thus potentially reducing block level contention.

Note Setting the PCTFREE parameter too high results in wasted block space and can increase the number of blocks that are required to store the same amount of data. It can also result in having to fetch more blocks that might otherwise be necessary.

Buffer caches

To provide separate buffer caches to help keep certain tables cached, and prevent their blocks from being aged out, you may consider allocating space to other buffer caches and then designate specific tables and indexes to those non-default buffer caches.

Block sizes

The default Oracle block size is 8K. If you are experiencing excessive row lock contention or block contention on the IN_OSM_TREE, IN_WF_QUEUE and IN_LOCK tables, all of which are small tables with small rows sizes, you can consider moving those tables and their respective indexes into 2K tablespaces. Moving the tables and indexes may help to reduce the level of contention on those tables.

It may also be beneficial to place larger tables, which are often a result of sequential multi-block reads, into a tablespace that uses a larger block size. For example, placing the IN_DOC and IN_INSTANCE tables in a tablespace created with a 32k block size could improve performance for some queries by reducing the number of blocks required to be read and transferred.

Note If utilizing non-default buffer caches, ensure an appropriate amount of memory is allocated to those buffer caches to accommodate the objects being placed there. Be sure that you periodically evaluate the Oracle buffer pool statistics as well as the advisory statistics for the various memory components to ensure they are set to optimal values.

Custom Indexes

From time to time it may be necessary to create a custom index on a table in order to achieve maximum performance for certain features or queries. When evaluating the creation of a custom index be sure to consider the entire product and not just a specific query. There is overhead associated with indexes so care must be taken to ensure it collectively does more good than harm.

When tuning a query, be sure to set all the standard session parameters. In particular, ensure that NLS_COMP = LINGUISTIC and NLS_SORT = BINARY_CI to ensure that the optimizer considers the existing function-based indexes that are created as part of the INUSER schema.

For any indexes that are created, and include a character based column, the indexed column must be wrapped using the NLSSORT function to ensure that the optimizer will even consider the index for Perceptive Content sessions.

Example: NLSSORT("DOC_TYPE_ID",'nls_sort=''BINARY_CI''')

Schemas

The following schemas are automatically created during execution of the full DDL script if they do not already exist. (SunflowerORA.sql or SunflowerORA-AdvancedTS.sql)

INUSER

The INUSER user/schema owns the Perceptive Content database objects and is the primary user and schema. The schema consists of several hundred tables, indexes, and constraints.

Minimum Privileges: GRANT CONNECT, RESOURCE, SELECT ANY DICTIONARY TO INUSER;

INEMUSER

INEMUSER user/schema- can be used to interface with the Perceptive Content database with limited permissions that are isolated to the IN_EXTERN_MSG (External Messaging) tables. IN_EXTERN_MSG, IN_EXTERN_MSG_PROP, IN_EXTERN_MSG_GROUP, IN_EXTERN_MSG_SEQ.

Minimum Privileges: GRANT CREATE SESSION TO INEMUSER;

Schema Version

You can determine the Perceptive Content database schema version by checking the PRODUCT_VERSION column of the IN_PRODUCT_MOD_HIST table using the following query:

```
SELECT * FROM INUSER.IN_PRODUCT_MOD_HIST ORDER BY MOD_TIME DESC;
```

The example below shows the schema version was initially created on schema version 7.5.0.0 then later upgraded to 7.7.0.0, which is the current version.

Example Results:

```
PRODUCT_MOD_ID           PRODUCT_VERSION   MOD_TIME                    PRODUCT_MOD_NOTES
------------------------ ----------------- --------------------------- -------------------
2100988013_1728017700    7.7.0.0           2021-06-18 17:32:54.480000   UTC
2100988013_1728017500    7.5.0.0           2020-02-12 16:27:45.153000   UTC
```

The INUSER schema has two possible configurations with respect to tablespaces and the location of each table and index within the tablespaces. The basic, standard configuration only consists of two tablespaces called DATA (for tables) and INDX (for indexes). The advanced configuration contains 116 different tablespaces, 58 for tables and 58 for indexes. Tables and indexes are strategically located across the different tablespaces to facilitate the distribution of IO across multiple, physical devices.

If the advanced tablespace schema has been implemented, the product_mod_id and product_version columns, returned by the query above, will have an “a” appended to the version number as show in the example below.

Example Results for Advanced Tablespace Configuration:

```
PRODUCT_MOD_ID           PRODUCT_VERSION   MOD_TIME                    PRODUCT_MOD_NOTES
------------------------ ----------------- --------------------------- ------------------
2100988013_1728017700a   7.7.0.0a          2021-06-18 17:32:54.480000   UTC
2100988013_1728017500a   7.5.0.0a          2020-02-12 16:27:45.153000   UTC
```

Tables

The INUSER schema contains nearly 400 tables. All table names are prefixed with “IN_”. Table names are also named accordingly so that the name includes distinguishing initialism to help identify the specific feature of the product for which the table corresponds to. For example a table prefixed with “IN_WF_” is used by the WorkFlow related features. Some tables will remain empty if the related features they support are not being used.

You can move individual tables into different tablespaces as necessary for general maintenance, to help offload I/O between database files or to make use of different block sizes.

Some of the larger tables will include the following list of tables:

```
IN_DOC
IN_DOC_KW
IN_INSTANCE
IN_INSTANCE_PROP
IN_LOGOB
IN_LOGOB_SUBOB
IN_PHSOB
IN_PHSOB_SUBOB
IN_SUBOB
IN_SUBOB_ANNOT
IN_SUBOB_ANNOT_COOR
IN_SUBOB_ANNOT_FONT
IN_SUBOB_ANNOT_TEXT
IN_VERSION
IN_VERSION_SUBOB
IN_WF_ITEM
IN_WF_ITEM_ARCH
IN_WF_ITEM_HIST
IN_WF_ITEM_HIST_ARCH
IN_WF_ITEM_QUEUE_HIST
IN_WF_ITEM_QUEUE_HIST_ARCH
```

Constraints

The Perceptive Content database uses all the standard constraint types to enforce uniqueness and referential integrity including Primary Keys, Foreign Keys, Unique, and Not Null check constraints.

Every table in the INUSER schema has a primary key constraint.

Primary key constraints are prefixed with PK_ and foreign key constraints are prefixed with FK_. All other constraints utilize system-generated names.

Certain columns, such as date fields, are populated with default values.

Indexes

The INUSER schema has just under 1,000 “function-based” indexes and just over 800 “normal” indexes.

In order to support the primary key constraints as well as efficient lookups used to facilitate referential integrity and data consistency checks “normal” indexes are created to support these constraints and are prefixed with either “PK_” or “FKCS_”.

Any other “normal” indexes that exist only contain columns that have a number or timestamp datatype. All other indexes, that include character-based columns, will be created as function-based indexes that utilize the nlssort function that matches the session level settings.

Case Insensitive (CI) searching has been implemented for the entire INUSER schema. To improve the performance of case insensitive (CI) searches, function-based indexes are created on certain character-based columns using the NLSSORT function along with BINARY_CI.

In addition to the “normal” PK indexes, function-based, unique indexes are created to support Case Insensitive uniqueness for each table and are prefixed with “PKCI_”.

You can move individual indexes into different tablespaces as necessary for general maintenance, to help offload I/O between tablespaces or to make use of different block sizes for tables.

Columns / Datatypes

Most of the columns are defined as VARCHAR2, TIMESTAMP(6) and NUMBER data types.

The following tables also contain LOB columns.

IN_AUDIT_DETAIL. DETAIL_VALUE column is defined as a CLOB.

IN_LIC_PACKAGE_CLAIM. CLAIM_DATA column is defined as a BLOB.

IN_MESSAGE. MESSAGE_BODY column is defined as a BLOB.

IN_SCRIPT.DATA column is defined as a CLOB.

Hidden Columns - Note that for every column that is part of a function-based index, a hidden column is automatically added to the table by the Oracle database. These columns will have system-generated names that start with SYS_NC000 and will have a RAW datatype. These columns are not controlled by or referenced by the Perceptive Content application.

Case insensitive schema implementation

Oracle is case sensitive by default. As part of the Oracle Globalization Support architecture, which was previously known as National Language Support (NLS), it provides linguistic sorting and matching along with a wide array of other locale specific configurations and formatting options. Starting with Oracle 10g Release 2 (10.2) the LINGUISTIC option was added to the NLS_COMP parameter in order to avoid having to use the NLSSORT or UPPER/LOWER functions in SQL statements.

When NLS_COMP is set to LINGUISTIC all SQL operations compare character values based on the collation specified in the NLS_SORT parameter. When NLS_SORT is set to BINARY_CI, it designates a collation that is accent-sensitive and case-insensitive. Together, the two session parameters, NLS_SORT and NLS_COMP, determine the rules by which character type data is compared and sorted.

The various NLS related parameters can be set at the database level during database creation or set and modified at the instance level using ALTER SYSTEM commands or at the session level using ALTER SESSION commands. Session level settings take precedence followed by instance and lastly database.

Session considerations

Support for case insensitive matching and sorting for Perceptive Content is currently accomplished by setting the NLS_COMP and NLS_SORT session parameters as follows, which results in all collation-sensitive SQL operations using binary values for collating and comparison while ignoring character case.

```
ALTER SESSION SET NLS_COMP=LINGUISTIC;
ALTER SESSION SET NLS_SORT=BINARY_CI;
```

All sessions established by the Perceptive Content application to an Oracle database set the following session level NLS parameters:

Oracle Session NLS Parameters

```
ALTER SESSION SET NLS_TIMESTAMP_TZ_FORMAT = 'RRRR-MM-DD HH24:MI:SS.FF TZR';
ALTER SESSION SET NLS_TIMESTAMP_FORMAT = 'RRRR-MM-DD HH24:MI:SS.FF';
ALTER SESSION SET NLS_TIME_FORMAT = 'HH24:MI:SS.FF';
ALTER SESSION SET NLS_DATE_FORMAT = 'RRRR-MM-DD';
ALTER SESSION SET NLS_COMP=LINGUISTIC;
ALTER SESSION SET NLS_SORT=BINARY_CI;
```

The various NLS parameters that are defined at each level can be viewed by selecting from the following Oracle views:

- NLS_DATABASE_PARAMETERS
- NLS_INSTANCE_PARAMETERS
- NLS_SESSION_PARAMETERS

Example:

```
INUSER@INOW SQL> SELECT * FROM NLS_SESSION_PARAMETERS;

PARAMETER                      VALUE
------------------------------ ------------------------------
NLS_LANGUAGE                   AMERICAN
NLS_TERRITORY                  AMERICA
NLS_CURRENCY                   $
NLS_ISO_CURRENCY               AMERICA
NLS_NUMERIC_CHARACTERS         .,
NLS_CALENDAR                   GREGORIAN
NLS_DATE_FORMAT                RRRR-MM-DD
NLS_DATE_LANGUAGE              AMERICAN
NLS_SORT                       BINARY_CI
NLS_TIME_FORMAT                HH24:MI:SS.FF
NLS_TIMESTAMP_FORMAT           RRRR-MM-DD HH24:MI:SS.FF
NLS_TIME_TZ_FORMAT             HH.MI.SSXFF AM TZR
NLS_TIMESTAMP_TZ_FORMAT        DD-MON-RR HH.MI.SSXFF AM TZR
NLS_DUAL_CURRENCY              $
NLS_COMP                       LINGUISTIC
NLS_LENGTH_SEMANTICS           BYTE
NLS_NCHAR_CONV_EXCP            FALSE
```

You can also use SYS_CONTEXT to view the various NLS and other parameters defined for your session within the USERENV context namespace.

SYS_CONTEXT

```
INUSER@INOW SQL> SELECT SYS_CONTEXT('USERENV', 'NLS_SORT') FROM DUAL;

SYS_CONTEXT('USERENV','NLS_SORT')
-----------------------------------
BINARY_CI
```

Schema considerations

When setting NLS_COMP to LINGUISTIC and NLS_SORT to BINARY_CI for the session there are several schema related considerations that become necessary for performance and operability. In order to maintain optimal performance, every index that is created on a character based column requires the NLSSORT() function that uses the same NLS_SORT value as the session (BINARY_CI) if it is to be considered by the optimizer when creating the explain plan for a query that is predicating on the indexed column(s). These type of indexes are called function-based indexes or linguistic indexes. Note that this only applies to character based columns and does not apply to numeric or date and time columns.

From an operability perspective, we cannot create primary key or unique constraints using functions, including the NLSSORT() function, so this necessitates the creation of "normal" primary key constraints to accommodate being referenced by foreign key constraints. With regard to the Perceptive Content schema, this results in all of the tables in the INUSER schema having a "normal" (non function-based) primary key constraint as well as a function-based unique index which is used to further enforce the uniqueness of the column in a case insensitive manner.

Using the IN_DOC table as an example, the following demonstrates the various constraints and indexes that we create to support a schema wide case insensitive database environment. This strategy applies to all the tables in the Perceptive Content database (INUSER schema).

Primary Key (PK) Constraint

We create the normal PK constraint which facilitates the creation of the FK constraints that reference the DOC_ID column. This is necessary because we cannot create a PK or unique constraint based on a function like NLSSORT().

```
ALTER TABLE INUSER.IN_DOC ADD CONSTRAINT PK_DOC PRIMARY KEY (DOC_ID)
  USING INDEX TABLESPACE INDX;
```

Foreign Key (FK) Constraint

We can now create foreign key constraints that reference the primary key constraint to maintain and document schema referential integrity.

```
ALTER TABLE INUSER.IN_VERSION ADD CONSTRAINT FK_V_DOC_ID FOREIGN KEY (DOC_ID)
  REFERENCES INUSER.IN_DOC(DOC_ID);
```

Foreign Key Case Sensitive (FKCS) Index

In order to maintain the performance of referential integrity checks as data is inserted, updated, or deleted we must also create normal indexes on the column(s) involved in the foreign key constraint where the leading column(s) of the key match the columns used by the FK constraint.

```
CREATE INDEX INUSER.FKCS_V_DOC_ID ON INUSER.IN_VERSION
(
    DOC_ID ASC
) TABLESPACE INDX STORAGE (INITIAL 64K NEXT 64K PCTINCREASE 0);
```

Primary Key Case Insensitive (PKCI) Unique Index

In order to enforce the case insensitive uniqueness for a column we must also create a unique index on the PK column(s) using the NLSSORT() function that specifies the same NLS_SORT setting as our session (BINARY_CI).

```
CREATE UNIQUE INDEX INUSER.PKCI_IN_DOC ON INUSER.IN_DOC
(
    NLSSORT("DOC_ID",'nls_sort=''BINARY_CI''') ASC
) TABLESPACE INDX STORAGE (INITIAL 64K NEXT 64K PCTINCREASE 0);
```

Function-Based Indexes

With the supporting constraints and indexes above, we then also create our remaining indexes to support our database queries and these must be created as function-based indexes if any of the columns include character base columns. Note that this is only a partial list of the indexes on the IN_DOC table.

```
CREATE UNIQUE INDEX INUSER.DOC_IDX2 ON INUSER.IN_DOC
(
    NLSSORT("DRAWER_ID",'nls_sort=''BINARY_CI''') ASC,
    NLSSORT("FOLDER",'nls_sort=''BINARY_CI''') ASC,
    NLSSORT("TAB",'nls_sort=''BINARY_CI''') ASC,
    NLSSORT("F3",'nls_sort=''BINARY_CI''') ASC,
    NLSSORT("F4",'nls_sort=''BINARY_CI''') ASC,
    NLSSORT("F5",'nls_sort=''BINARY_CI''') ASC,
    NLSSORT("DOC_TYPE_ID",'nls_sort=''BINARY_CI''') ASC,
    NLSSORT("LEGACY_UNIQUENESS",'nls_sort=''BINARY_CI''') ASC
) TABLESPACE INDX STORAGE (INITIAL 64K NEXT 64K PCTINCREASE 0);

CREATE INDEX INUSER.DOC_IDX11 ON INUSER.IN_DOC
(
    NLSSORT("DRAWER_ID",'nls_sort=''BINARY_CI''') ASC,
    CREATION_TIME ASC
) TABLESPACE INDX STORAGE (INITIAL 64K NEXT 64K PCTINCREASE 0);

CREATE INDEX INUSER.DOC_IDX5 ON INUSER.IN_DOC
(
    NLSSORT("DOC_TYPE_ID",'nls_sort=''BINARY_CI''') ASC,
    NLSSORT("DOC_ID",'nls_sort=''BINARY_CI''') ASC
) TABLESPACE INDX STORAGE (INITIAL 64K NEXT 64K PCTINCREASE 0);

CREATE INDEX INUSER.DOC_IDX81 ON INUSER.IN_DOC
(
    NLSSORT("FOLDER",'nls_sort=''BINARY_CI''') ASC
) TABLESPACE INDX STORAGE (INITIAL 64K NEXT 64K PCTINCREASE 0);
```

Oracle Case Insensitive Demonstration

```
INUSER@INOW SQL> alter session set nls_comp=binary;

Session altered.

INUSER@INOW SQL> alter session set nls_sort=binary;

Session altered.

INUSER@INOW SQL> select doc_id, drawer_id, doc_type_id,
folder from in_doc where folder like 'DDDDD%';
```

```
no rows selected


Execution Plan
----------------------------------------------------------
Plan hash value: 1179247437

--------------------------------------------------------------------------------
| Id  | Operation         | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |        |     1 |    90 |    35   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| IN_DOC |     1 |    90 |    35   (0)| 00:00:01 |
--------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("FOLDER" LIKE 'DDDDD%')


INUSER@INOW SQL> alter session set nls_comp=linguistic;

Session altered.

INUSER@INOW SQL> alter session set nls_sort=binary_ci;

Session altered.

INUSER@INOW SQL> select doc_id, drawer_id, doc_type_id,
folder from in_doc where folder like 'DDDDD%';

DOC_ID                  DRAWER_ID               DOC_TYPE_ID             FOLDER
----------------------- ----------------------- ----------------------- --------------
---------------------------
321Z279_002Z7QNMW000028 321YY1W_0001EB73S00002D 321Z04T_001F8KR5F00002W ddddd
321Z279_002Z7VNMW000015 321YY1W_0001EB73S00002D 321Z04T_001F8KR5F00002W dddddd
321Z279_002Z7MNMW000061 321YY1W_0001EB73S00002D 321Z04T_001F8KR5F00002W ddddddd
321Z279_002Z7MNMW000068 321YY1W_0001EB73S00002D 321Z04T_001F8KR5F00002W dddddddd
321Z279_002Z7MNMW00006C 321YY1W_0001EB73S00002D 321Z04T_001F8KR5F00002W ddddddddd
321Z279_002Z7LNMW000046 321YY1W_0001EB73S00002D 321Z04T_001F8KR5F00002W dddddddddd

6 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 1393760941

--------------------------------------------------------------------------------------------
-----------
| Id  | Operation                           | Name      | Rows  | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------------------------------
-----------
|   0 | SELECT STATEMENT                    |           |     5 |   545 |     1   (0)|
00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| IN_DOC    |     5 |   545 |     1   (0)|
00:00:01 |
|*  2 |   INDEX RANGE SCAN                  | DOC_IDX81 |     5 |       |     1   (0)|
00:00:01 |
```

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

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access(NLSSORT("FOLDER",'nls_sort=''BINARY_CI''')>=HEXTORAW('646464646400') AND
              NLSSORT("FOLDER",'nls_sort=''BINARY_CI''')<HEXTORAW('646464646500'))
```

Case Insensitive summary

This solution creates an implementation that is fully case insensitive for our database sessions and affects the entire schema. Unfortunately, it also requires more indexes than would otherwise be necessary since we must create normal (non function-based) indexes to support database constraints (PK, FK) and we must also create unique function-based indexes (PKCI) that are the case insensitive counterpart to our PK constraints/indexes to facilitate uniqueness in a case insensitive manner.

Database schema upgrades

The Perceptive Content database schema is upgraded by executing the appropriate incremental DDL scripts in the proper, sequential order until you have reached your target schema version number.

Alternatively, the IN_DB_UPGRADE package can also be used to upgrade the INUSER database schema and offers a number of benefits over the traditional database incremental scripts. Please reference the Perceptive Content Database Oracle Upgrade Guide for more details.

Database upgrade scripts can be found on the Support->Software Downloads->Perceptive Downloads page within the Customer Portal on the Hyland Community website.

Database connectivity and security

Database connections

The Perceptive Content application server utilizes self-managed connection pools that are configurable within the respective .ini files for the associated service. These connections are established when a service is started and will remain connected, in an idle state, until needed by the service to handle any database requests.

Each database connection is tagged by the service so that the session contains the CLIENT_INFO and MODULE associated with the specific service. These identifying session attributes can be used when filtering connections during troubleshooting or performance tuning efforts.

Security

Database connections are established directly between the application server and the database, and the likelihood of a connection in the pool being hijacked and utilized in a malicious manner is relatively low if the network topology is properly isolated and patched, and all network traffic is encrypted and transmitted using secure protocols. Consider encrypting network traffic to the database using SSL. Please refer to the *SSL Security with an Oracle Database Best Practices* documentation or the Oracle provided documentation.

ODBC drivers

Beginning with Perceptive Content 7.3.0 (EP1), you must install and configure the Oracle native ODBC and Instant Client on the Perceptive Content application server. Refer to the *Perceptive Content Database Driver Installation and Configuration Guide* for details. Older versions of Perceptive Content use the DataDirect ODBC drivers to establish connectivity between the application server and the database.

Quick Reference Guide

The following checklist provides a quick reference for your Oracle database. It is not meant to be an all inclusive task list but is provided to help outline some of the areas where consideration should be given when planning and creating a new Oracle environment. Be sure to reference Oracle provided documentation for complete details and guidance.

✓	**Task**
	Check the *Perceptive Content Technical Specifications* for your release of Perceptive Content for the supported database versions.
	Review database features for the different Oracle editions and choose the appropriate edition of Oracle to use in your environment.
	Select an OS that can be effectively managed by supporting staff as well as factoring in operating costs and performance.
	Configure an isolated network topology using encryption and secure protocols. Reserve IP addresses to accommodate your implementation of Oracle with consideration to GRID (Clusterware/ASM) configurations, including DNS entries for SCAN addresses, private cluster interconnect traffic, local listeners, etc.
	Design the underlying storage system for the Oracle database with consideration for shared storage, if using GRID, and consideration for the binaries, system database files, application database files, fast recovery area, undo, temporary tablespaces, redo logs, archived logs, and backups. Ensure the storage solution has proper capacity for expected growth based on short term and long term capacity planning.
	Consider using Oracle Automatic Storage Management (ASM) for database storage management.
	Check that all Oracle installation prerequisites have been met or exceeded.
	Install Oracle binaries using an “Optimal Flexible Architecture” configuration as described in Oracle installation guides.
	After installing Oracle, apply the latest recommended Oracle patches for security and bug fixes.
	If implementing SSL for secure network traffic and data transmission then configure the Oracle wallet using the orapki utility. Have certificate requests signed using a trusted Certificate Authority. Make the necessary changes to the Oracle SQLNET files to reflect the wallet location, SSL version, cipher suites, TCPS protocol and port number changes, and distinguished names used by the certificates.
	Create the INOW database using the WE8MSWIN1252 database character set.
	Multiplex Control Files and Redo Logs.

	Periodically evaluate log switches and adjust the number and sizes of the redo logs as needed. Ensure all redo logs are the same size. Set redo logs to switch about every 15 minutes during peak hours.
	Configure the database to run in archivelog mode. Utilize multiple archive destinations if possible, including the fast recovery area.
	Set total SGA memory size to a minimum of 10GB.
	Set PGA aggregate target to 2GB.
	Periodically evaluate the Oracle buffer pool statistics as well as the advisory statistics for the various memory components to ensure they are set to optimal values.
	Configure HugePages if the OS supports it.
	Set PRE_PAGE_SGA = TRUE if the OS supports it.
	Set LOCK_SGA = TRUE if the OS supports it.
	Set PROCESSES to at least 600 or higher depending on the expected number of database connections.
	Use the Dedicated Server Mode for the INOW database.
	Set UNDO_RETENTION to 9000 and allocate sufficient storage to the UNDO tablespace to accommodate the retention period based on the Undo Advisor recommendations.
	Gather initial server statistics and then disable server statistics gathering unless server config changes
	Gather schema statistics periodically using the recommendations outlined above.
	Schedule daily online database backups using RMAN to disk or tape. If you backup to disk, use a third-party media management program to transfer RMAN backups to tape.
	Schedule frequent archivelog backups, if running in archivelog mode.
	Set up daily data pump exports of your database if feasible.
	Change the default passwords for all users
	Increase the AWR retention period to 30 days.
	Review Oracle AWR and ADDM reports periodically.
	Create baseline reports to document expected, good performance which can be used to compare against future reports when troubleshooting performance issues.
	Perform periodic database restore and recovery testing to ensure disaster recovery preparedness.