ORACLE JD EDWARDS ENTERPRISEONE

ORACLE DATABASE 12c

Performance Characterization of Oracle JD Edwards EnterpriseOne with Oracle Database In-Memory

EnterpriseOne provides real time analytical functions with its new financial reconciliation process and adds value to its already rich features

ORACLE WHITE PAPER | MARCH 2015 | VERSION 3

ORACLE

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Table of Contents

Executive Summary

Oracle Database In-Memory transparently accelerates analytic queries by orders of magnitude, enabling real-time business decisions. Using Database In-Memory, JD Edwards EnterpriseOne customers can instantaneously run analytics and reports that previously took hours or days. Businesses benefit from better decisions made in real-time, resulting in lower costs, improved productivity, and increased competitiveness.

Inspired by the capabilities offered by Oracle Database In-Memory, Oracle engineers incorporated two new features, real-time summarization and real-time financial reconciliation into the software. These product enhancements result in new features that transform formerly time-consuming batch processes into real-time analytic applications. Oracle conducted performance tests with JD Edwards EnterpriseOne and Oracle Database In-Memory. Many test cases demonstrated astounding results, some with orders of magnitude - 10X, 100X, and even over 1,000X faster.

This paper describes three usage patterns, backed by performance test cases that benefit from running JD Edwards EnterpriseOne with Oracle Database In-Memory:

» **Real-time Operational Analytics.** In this usage pattern Database In-Memory enables users to execute analytic queries against the transactional database tables. This paper describes test cases that resulted in queries that ran *684 times faster* with Oracle Database In-Memory.

» **Real-time Summarization.** In this usage pattern, the real-time summarization applications with Database In-Memory enabled users to show summary values of grid columns for amounts in real time without the need to perform a 'Go to end' action followed by an export to an external spreadsheet or run a custom batch program. This paper describes the test cases that range from *2 to 2994 times faster*.

» **Real-time Financial Reconciliation.** In this usage pattern, Oracle has reengineered the process by which customers reconcile financial records, transforming it from lengthy batch jobs into real-time, interactive applications. This paper describes test cases that range from *50 to 272 times faster*.

Because Oracle Database In-Memory can be implemented with *no changes to JD Edwards EnterpriseOne applications*, customers can begin their own discovery of use cases where real-time analytics can transform their business processes. Meanwhile, the success of initial product enhancements to JD Edwards EnterpriseOne lays the groundwork for future enhancements that further capitalize on the capabilities of Oracle Database In-Memory.

The ability to easily perform real-time data analysis together with real-time transaction processing on all existing applications enables JD Edwards EnterpriseOne customers to transform into Real-Time Enterprises that quickly make data-driven decisions, respond instantly to customer demands, and continuously optimize all key processes.

Introduction: Why Do I Need an In-Memory Database?

Oracle's JD Edwards EnterpriseOne is an enterprise resource planning (ERP) software system that combines business value, standards-based technology, and deep industry experience into a business solution. Enterprises of all sizes, industries, and geographies use JD Edwards EnterpriseOne to capture, manage, and analyze the data that makes their businesses compete and succeed.

As is typical of ERP systems, JD Edwards EnterpriseOne provides a set of software modules that allows end users to capture and report on financial transactions, inventory, manufacturing operations, projects, customers, employees, and many other activities of a modern enterprise. As part of their daily activity end users need to query data from the system to support traditional ERP transactions, such as invoicing a customer or shipping a product. However, in an increasingly competitive business environment, executives, managers, and empowered employees also need data from the system to support their decision-making, such as analyzing sales trends or responding to unplanned events. As businesses grow and as time passes, the data collected within the system can grow at an astounding rate.

To keep the enterprise operating efficiently, administrators are charged with the responsibility of tuning the hardware, ERP software, and database to run at optimal performance. Unfortunately, administrators are pressured from both sides: the system must be continually tuned as human and nonhuman users exercise the database with transactional data; meanwhile, users, managers, and executives conceive of new queries, dimensions, and usage patterns, which the system may not be tuned to satisfy. How are system and database administrators to deal with incessant data expansion and at the same time fulfill new requests for that data? A variety of approaches are available, but each comes with its compromises and trade-offs:

- **Batch jobs.** If an end user's query for data returns too slowly to be practical within an interactive context, the traditional solution is to write a batch report that runs off-line. While this solution does not actually increase the speed at which the system returns the data, at least the user is free to go on with other tasks while the system batch job works in the background to return the data. If the use case is repetitive and reusable, for example a financial report that runs at the end of every month, then investment in developing the batch report may be justified. But if the data request is an infrequent case, the cost does not justify the "one-off" request.

- **Database indexes.** Database administrators can create indexes over the database tables, which can accelerate the database's ability to return data to the application. However, an index is often applicable to a specific or narrow set of usage patterns. Creating an index requires the engagement of a capable database administrator, and the index itself consumes system resources, such as CPU, memory, and disk space. As users request more and more multidimensional data, and as the set of indexes grows, the system can actually decrease its efficiency to complete database requests to a point of diminishing return.

- **Just say “no.”** Sometimes an end user has an unanticipated “ad-hoc” need for data. Some of these “ad-hoc” needs for data stem from unanticipated business situations that require a company to make educated, informed decisions very quickly in order to reduce risk, remain competitive, retain customer satisfaction, just to name a few situations. Being unpredictable, it is not possible to predetermine the need for an index over that data or a report to fetch it asynchronously in a batch process. If that data resides in a very large table, the database request might not just create an additional load on the system, but also may not return the result to the interactive user before an application time-out occurs. And the “fix”? Typically it is to use an application or database security task to prevent users from running such queries. Yes, that approach protects the system and averts application failures, but it does little to satisfy the user’s original request for data.

- **Use another system.** When the data is needed to support analytical use cases, sometimes the ERP data is exported to an external data warehouse. In this process, the data can also be transformed from its transaction-optimized row-based format to a query-optimized columnar format. While data warehouses and business intelligence applications can unlock a wealth of value within the data, they do require additional resources, and the data within them can only be as current as the rate at which it is refreshed from the ERP system.

Oracle Database, and specifically the Database In-Memory option, provides an innovative new alternative that addresses these problems without the compromises. With the introduction of Database In-Memory, a single database can now efficiently support mixed workloads, delivering optimal performance for transactions while simultaneously supporting real-time analytics and reporting. This is possible due to a unique "dual-format" architecture that enables data to be maintained in both the existing Oracle row format, for OLTP operations, and a new purely in-memory column format, optimized for analytical processing. While customers may experience some performance benefit to their existing JD Edwards EnterpriseOne applications, the dramatic benefit of Database In-Memory will be realized in unlocking new analytical usage patterns over the same real-time OLTP data without modifying or impacting existing applications.

How JD Edwards EnterpriseOne Benefits from Oracle Database In-Memory

Oracle Database In-Memory allows transactional applications to coexist with analytical queries, transforming the applications that use them. Enabling the Database In-Memory feature with JD Edwards EnterpriseOne, either with unmodified applications or in conjunction with new product enhancements, allows businesses to gather and report analytic information interactively in real time which previously took hours to produce in a batch process. With these features, businesses have the ability to make real time decisions with real time information, thus reducing risks in their business. Companies also benefit from shortened business cycles, such as financial reconciliation, which lowers a company's overall TCO.

This document describes this real time computing and how it can transform the traditional batch EnterpriseOne processes into an interactive real time system. JD Edwards EnterpriseOne is an Enterprise Resource Planning (ERP) system in which customers have stored, over time, a tremendous amount of data into the EnterpriseOne tables. Oracle is now providing more analytical tools to unlock the power of this data. This new real time analytics environment can extend the ERP business model, allowing processes to be performed in real time, which previously took hours to perform in batch.

The benefits of using Database In-Memory with JD Edwards EnterpriseOne centers around three areas:

- Real-time operational analytics
- EnterpriseOne real-time summarization
- EnterpriseOne real-time financial reconciliation

Although the Database In-Memory feature can be applied to the EnterpriseOne application architecture unchanged, the opportunity to provide both Database In-Memory feature enablement and JD Edwards EnterpriseOne product enhancements is the subject of this document. The results, in a few cases, achieved greater than 1000x performance improvements.

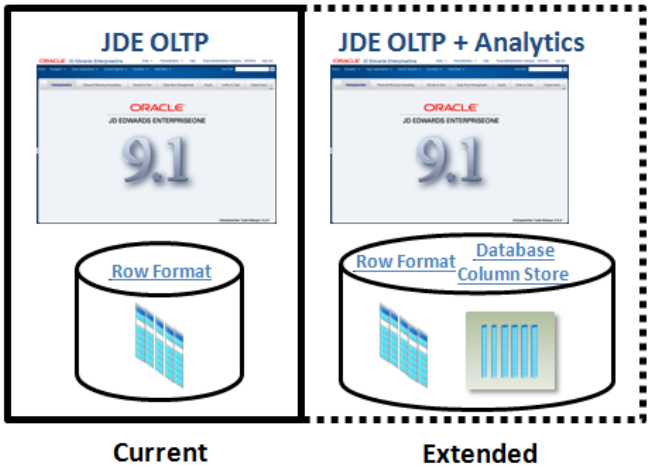

Figure 1: JD Edwards EnterpriseOne with Database In-Memory

Figure 1 illustrates the dual-format architecture enabling OLTP and analytics to be performed in EnterpriseOne from the same database. The addition of the database column store can extend the analytical capabilities to the current JD Edwards EnterpriseOne architecture.

This document reports the results of performance testing utilizing the Database In-Memory capabilities, along with those tests with additional product enhancements to the EnterpriseOne system, in order to greatly improve performance of the software and user experience. This combination will enable doing scans, joins, and aggregates calls much faster than performing traditional row format processing historically seen in the EnterpriseOne workloads.

The Database In-Memory feature discussed in this document is comprised of:

- A column store.
- Storing the column store in RAM memory (which is in compressed format, saving precious space in RAM, a key resource to the Oracle database).

Another important benefit to this database technology is that it can co-exist with the traditional row format technology. Thus, the Oracle optimizer can choose the best of both configurations to best suit the needs of the EnterpriseOne application.

Database In-Memory Column Store

The column store compression for EnterpriseOne represents the selected tables that were placed in the column store in a compressed format. The default compression level was used in all testing by specifying the 'MEMCOMPRESS FOR QUERY LOW' option when enabling EnterpriseOne tables for Oracle Database In-Memory. There are six levels of in-memory compression. The default compression level is optimized for query performance. The compression ratio for most of the EnterpriseOne tables is similar to other applications ranging from 10x-20x as reported in other Oracle literature. A number of factors contribute to storing the EnterpriseOne data in the column store, including:

- EnterpriseOne data is sparsely populated in many of the columns in the database. This empty space is highly compressible.
- Many of the values in EnterpriseOne data fields are repeated from row to row. Fields such as date, owner, and column fields, such as category codes, document types etc., have very low numbers of distinct values. When there are a low number of distinct values in a column, the data is highly compressible.
- The data set used for reporting results in this document was non-production testing data selected to represent an 'average' customer size.

Customers in specific vertical industries might experience a different compression ratio because of their unique business processes and data sets for their implementation of Database In-Memory column store.

EnterpriseOne and the Database In-Memory Feature

Row format will favor the EnterpriseOne transactional database requests and the compressed column store in memory can service extended analytical queries to the database. **Figure 2** below illustrates the Database In-Memory feature. This feature has the distinct advantages of being transparent to the EnterpriseOne application and having the ability to function independently.

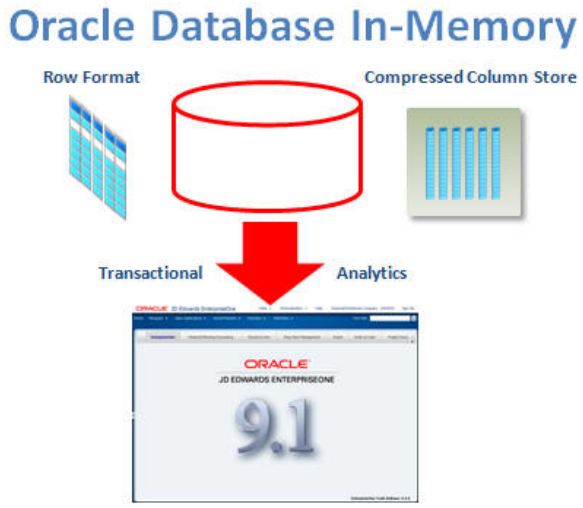

Figure 2: Database In-Memory Benefits

Figure 2 above also illustrates that both transactional and analytical benefits flow from a single Oracle database to the real time user interface of EnterpriseOne.

Extending EnterpriseOne Functionality

The ability of real time analytic functionality in an already powerful EnterpriseOne application can extend its current capabilities in this area to provide revolutionary change to the customers who use EnterpriseOne but have not unlocked its full potential for real time computing in the form of reports and real time processing. **Figure 3** below illustrates this transformation.

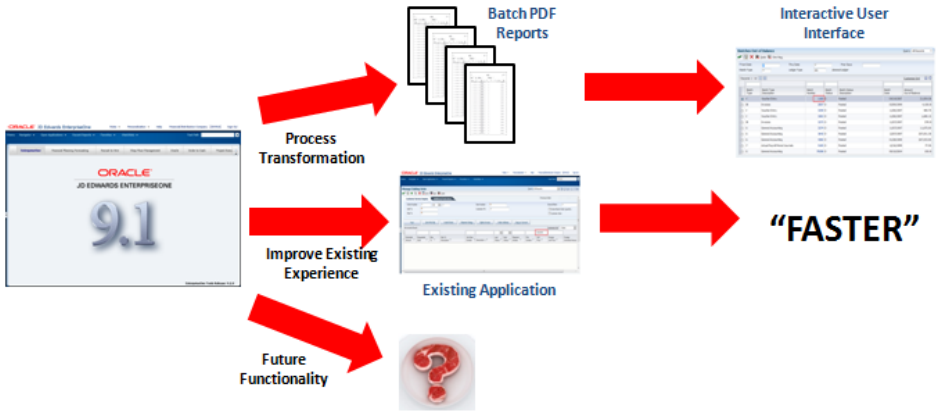

Figure 3: EnterpriseOne Real Time Computing

Figure 3 illustrates three areas where adopting the Database In-Memory features can extend the capabilities of the EnterpriseOne application:

1. **Process Transformation** - The inclusion of the Database In-Memory into the financial integrity reports and the design of a new interactive application that performs this function replacing a batch process with an interactive user process that can perform at levels of efficiency as to be considered real time.
2. **Improve Existing Experience** - Traditional slow running grid queries may find the unexpected performance boost by using the Oracle Database In-Memory column store. Ad-hoc queries performed in a number of interfaces may find unexpected value because the Database In-Memory column store can perform non-indexed database queries much faster than in the historical row based model.
3. **Future Functionality** - Implementing Database In-Memory provides an architectural foundation of capabilities to the EnterpriseOne application. The analytics capability provides a foundation for future development for both the Oracle delivered applications and those customized by the customer for EnterpriseOne.

Simply implementing the Database In-Memory column store feature with EnterpriseOne and placing random EnterpriseOne tables into it is not likely to translate to a dramatic benefit to traditional ERP transactional applications. Apart from the newly designed real-time financial reconciliation and summarization processes and the known benefit to ad-hoc queries, the EnterpriseOne performance specialist in conjunction with database administration support should work together to find those areas of the existing application that can provide benefit.

There will be many references to the column store Database In-Memory feature in this document. For a detailed overview to the Database In-Memory functionality, see:

Reference (October 2014): White Paper: Oracle Database In-Memory

http://www.oracle.com/technetwork/database/in-memory/overview/twp-oracle-database-in-memory-2245633.html?ssSourceSiteId=ocomen (Audience: Database administrators and performance specialist)

EnterpriseOne Applications Tested with Database In-Memory

The EnterpriseOne applications discussed in this document include general “ad hoc” queries from standard JD Edwards EnterpriseOne modules, new features that summarize grid data, and new interactive integrity reports. For purposes of this document these three areas will now be referred to as: real-time operational analysis, real-time summarization, and real-time financial reconciliation.

The performance test cases in this document were conducted with JD Edwards EnterpriseOne utilizing Oracle Database 12c Enterprise Edition with Oracle Database In-Memory. For detailed information regarding the test environment and EnterpriseOne required versions, see the Appendix.

Real-time Operational Analysis

In the real-time operational analysis use cases, an end user needs to query for data from the system, but the query is an unanticipated and perhaps “irregular” need for data. This pattern is sometimes referred to as an “**ad-hoc”** query. Because the query is not a regular recurring case, it is impossible to anticipate the use case with a specialized application, report, or database index. Therefore, the EnterpriseOne software application will perform a non-indexed database search for the result. Traditionally, for good performance, software applications restrict certain grid column searches because the resulting request issued to the database would be over a non-indexed column.

For example, consider the case with the P42101 (Sales Order Entry) application where the search is for a specific amount in the grid field, as in Figure 4. **Figure 4** shows a search for a specific unit price in all of the sales orders in the database. In this case however, the unit price grid field is a non-indexed column in the database.

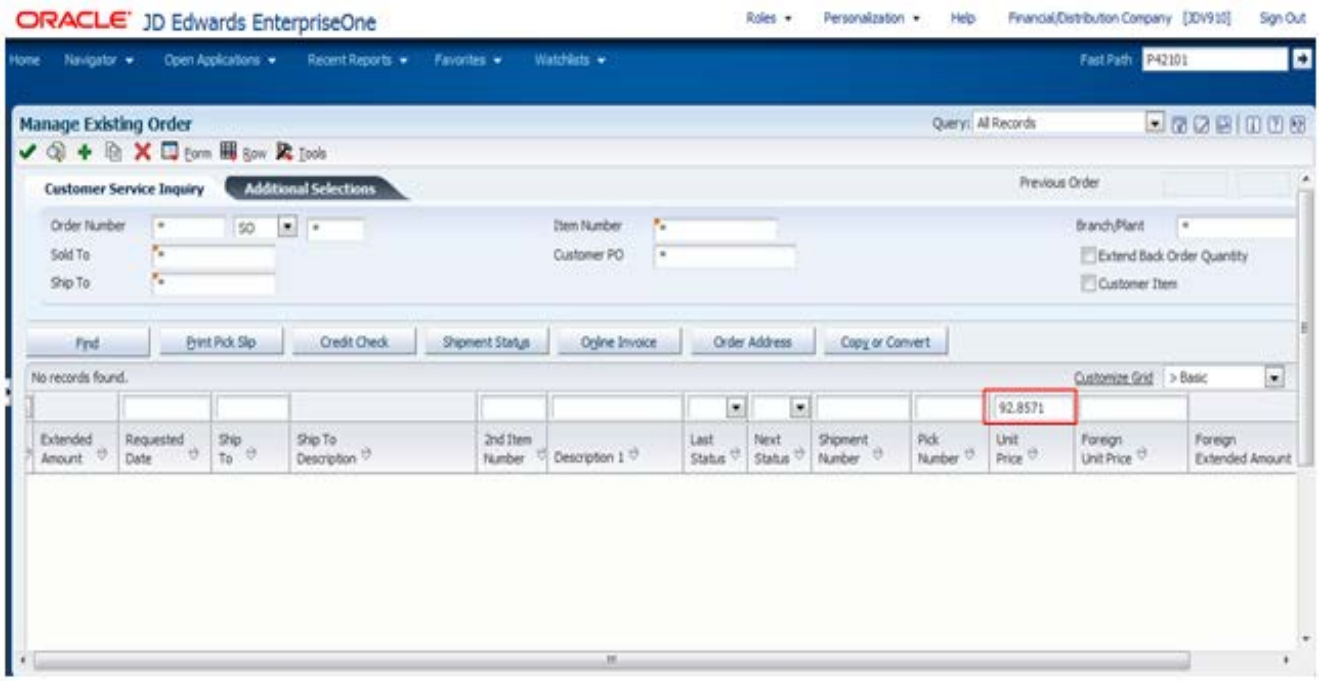

Figure 4: Ad-hoc Query Scenario Sales Order Entry

The problem is the end user does not have a sense as to which grid fields are a database indexed or non-indexed column and most likely they are not concerned with this low level of information. What they are concerned with is the application’s ability to return the information they are requesting in a reasonable amount of time.

A traditional approach to resolving this problem is either through end user training to not perform these types of queries,, leverage query security to require indexed columns to be included in the query, grey out these columns in the grid so as to not allow the end user to

enter informational requests using these fields, or to create a batch report to provide the end user with the result of this type of information request. The traditional solutions are essentially sidestepping the real issue; performance of the database request for the information.

Many poor user interactive experiences are attributed to poor performance in the database request. Database in-memory column store can address this issue when the query is found to be over a non-indexed column that is on a large EnterpriseOne table.

Real-time Operational Analysis of Non-Indexed Columns

Non-indexed columns are very inefficient and slow compared to those searches using a primary key or other indexed column because the database must perform a slow full scan of the entire table to retrieve the results.

The Oracle optimizer determines whether to perform a full-scan of the table or use a database index. By implementing the Oracle database in-memory column store, where the row and column formats are both available to the Oracle optimizer to make this decision, database requests that historically have performed poorly because on non-indexed columns can now perform efficiently in real time. In the case of the query in **Figure 4**, since the dataset returned is small in comparison to the number of records processed, a huge gain was seen in the performance of the database request because the request used the column store and in return added to the end user experience.

With the use of the Oracle Database In-Memory store, customers can benefit in a number of ways related to real-time operational analysis. First, customers will be able to drop some indexes and continue to retrieve the same information in a similar amount of time. Second, customers may be able to eliminate the need to create and maintain additional customizations. Third, customers can quickly find specific information when performing "ad-hoc' queries that have no related operational index. Finally, customers may be able to eliminate the need to create additional operational indexes.

Real-time Operational Performance Test Results

The purpose of this section is to describe the testing approach which shows the power of the Oracle Database In-Memory column store when performing interactive user requests over non-indexed columns where the historical performance was slow or timed out. It is important to note that only the Oracle Database In-Memory column store was implemented and no product enhancements have been made to the EnterpriseOne applications. All performance gains and superior end user experience in these examples are fully attributed to the Oracle Database in memory feature.

The results in **Table 1** for the ad-hoc query testing were achieved under the following conditions:

- There were no changes to the applications programs.
- The Oracle Database 12c column store and compression feature was implemented.

The tests were conducted on non-indexed queries where the result set is a small set of records compared to the total number of records in the table.

The following are four representative use cases for real-time operational analysis. See **Table 1** for the results.

Item Master Query (P4101)

Sample Use Case: A sales clerk is on the phone with a customer and the customer wants to order an item but does not know the item number or the specific name of the item. The sales clerk uses a wildcard search to quickly find all the items that meet a partial description of the item and reviews these with the customer. The customer then is able to identify the item they wanted for their order.

1. Login to the EnterpriseOne application.
2. Launch the Item Master application (P4101).
3. Perform a wildcard search (i.e. 'ITEM*') in the Search Text grid QBE field.
4. Click the 'Find' button.

Customer Ledger Query (P03B2002)

Sample Use Case: A customer calls in with regard to a shipment. The customer thinks that there were multiple orders in the shipment but cannot recall all of the order numbers but does have the shipment number. The customer service representative quickly finds three orders for the customer based on this shipment number without having to do any other searches.

1. Login to the EnterpriseOne application.
2. Launch the Customer Ledger application (P03B2002).
3. Perform a search for a select number (i.e. '789123') in the Shipment Number QBE field.
4. Click the 'Find' button.

Sales Order Query (P42101)

Sample Use Case: The sales manager discovers that a number of items were incorrectly priced at $92.15 per unit. She wants to see all sales orders entered with this unit price to review them. The sales manager quickly finds only one sales order out of all sales orders in the system with an item with a unit price of $92.15 and takes prompt action.

1. Login to the EnterpriseOne application.
2. Launch the Sales Order application (P42101).
3. Enter a numeric value (i.e. '92.15') into the Unit Price grid QBE field.
4. Click the 'Find' button.

Purchase Order Query (P4310)

Sample Use Case: The purchasing manager learns that a few months ago a purchasing clerk entered in a purchase order for the amount of $1,000 which should have been $10,000. The clerk cannot recall what order type they entered for the purchase order or the customer. The purchasing manager quickly searches for all types of orders with orders of $1,000 and is able to quickly identify the order in question.

1. Login to the EnterpriseOne application.
2. Launch the Purchase Order application (P4310).
3. Set the Order Type to '*', default is normally specified in IV processing option (i.e. 'OP').
4. Enter a numeric value (i.e. '1000') into the Amount grid QBE field.
5. Click the 'Find' button.

For the testing scenarios presented in **Table 1**, the values represent a 'first hit' timing. A first-hit is what an ad-hoc query would typically experience. First-hit, from a database perspective, means that the result set of information does not currently reside in the database buffer cache nor any other cache structures and so must be retrieved from disk, as in the case of row format, or the column store, as in the case of Oracle in-memory.

Other cache structures that might have recent result sets include the flash-cache storage RAM as is in the case of Exadata engineered systems. Thus what is being measured in these tests is a comparison between the Oracle in-memory column store and row based retrieval.

To further ensure the accuracy of the performance of the Oracle optimizer, complete Oracle database statistics were performed prior to the testing of the use cases.

EnterpriseOne Application	Table Queried	Number Records in Table	Records Returned	BASLN Time (s) INDEX FULL SCAN	BNCHMK InMemory Enabled Time (s) INMEMORY	BNCHMK InMemory Disabled Dropped Indexes Time (s) INDEX FULL SCAN	BNCHMK InMemory Enabled Dropped Indexes Time (s) INMEMORY	X-Times Faster (rounded)
Item Master Query (P4101)	F4101	923K	First-10	95.599	0.676	3.142	0.589	**162.31**
Customer Ledger Query (P03B2002)	F03B11	10M	3	342.010	0.624	16.158	0.594	**575.77**
Sales Order Query (P42101)	F4211	66M	1	325.661	0.438	21.670	0.476	**684.16**
Purchase Order Query (P4310)	F4301	512K	First-10	31.721	1.102	29.958	1.149	**27.61**

Table 1: EnterpriseOne Ad-Hoc Results

For each of the EnterpriseOne Applications in **Table 1** and **Table 2** for the Real Time Summarization discussion, there were two formal data points; the baseline and the benchmark. A brief discussion of the terms used in this section will help bring context to the results presented in this document.

Table Queried: An ad-hoc query centers around the database time spent searching for a result in a primary EnterpriseOne table. Typically, the larger the size of the table, the longer a non-indexes column search will take to perform the action.

Thus, the initial baseline timings will increase with increasing record counts. **Table 1** lists the number of records that were in the primary table for the tests results in this document.

Records Returned: EnterpriseOne has a feature to return the first 10 records by default with large queries. Table 1 illustrates varying number of records returned for the use cases tested. In two of the use cases, the result set returned the default 10 records; more records could be retrieved by initiating the 'Go to end' action.

Baseline (BASLN) Time: This value is the baseline metric of elapsed time (in seconds) as a point of reference. In the case of each of the EnterpriseOne applications, this represents the current EnterpriseOne functionality of the software application.

In the example of Sales Order Query (P42101), the result is 325.661 seconds. In a customer environment where the table is larger than the 66 million records used in this test, the results of the query is likely to time out and not return any information to the user.

Benchmark (BNCHMK): The metric of elapsed time (in seconds) for the benchmark is defined by the EnterpriseOne application product enhancements and/or implementation in conjunction with the Database In-Memory column store.

A benchmark is identical to a baseline in the EnterpriseOne user actions for the same application with the exception that the tests were performed with multiple Oracle configurations. The first Oracle configuration is with the in-memory column store enabled and having the EnterpriseOne indexes present (which is the 'out of the box' default). The second scenario is with the in-memory column store disabled and the EnterpriseOne indexes dropped. The testing with EnterpriseOne indexes dropped translates into dropping all of the EnterpriseOne indexes with the exception of the EnterpriseOne primary keys. EnterpriseOne primary key constraints maintain data integrity for the EnterpriseOne application and so they were not dropped for the purpose of this analysis.

Finally, the testing was performed with the Oracle Database In-Memory column store enabled and the EnterpriseOne indexes dropped, again with the exception of all unique primary key constraint indexes. The dropping of all the performance indexes in this testing was an extreme case to review the Oracle optimizer characteristics. Database administrators and EnterpriseOne application specialists should work together in a holistic approach when dropping indexes in a production environment.

X-Times Factor: This value represents the baseline elapsed time divided by the benchmark with the Oracle In-Memory feature enabled elapsed time and EnterpriseOne dropped indexes. The improvement factor is reported as rounded to two decimal places. For Sales Order Query, the X-Times Factor is (325.661/0.476) or 684.16 times faster.

INMEMORY and INDEX FULL SCAN: As part of the timing in Table 1, the Oracle optimizer execution plan is listed in the column headings. In all cases when the Oracle Database In-Memory was enabled for real-time operational analytics, the Oracle column store was chosen by the Oracle optimizer as the best execution of the SQL query. The 'INMEMORY' designation on the main table listed for each use case was found in the SQL execution plans. This was the case when both EnterpriseOne indexes were present and for the case in which only the primary key EnterpriseOne index was present, designated by the 'Dropped Indexes' column notation.

When the Oracle Database In-Memory feature was turned OFF, an 'INDEX RANGE SCAN' was found to be chosen by the Oracle optimizer. In the column where only the primary key EnterpriseOne index was present, the primary key 'INDEX RANGE SCAN' was the execution plan. In the case where all EnterpriseOne indexes were available to the Oracle optimizer, the execution plans still performed an 'INDEX RANGE SCAN' on the main EnterpriseOne table for the SQL query, but often used an index other than the primary key.

Real-time Operational Analysis Discussion

The results presented in Table 1 are too wide for interpretation without this section of the document for discussion. This discussion will be prioritized by the messages that are most important to the reader. The three important discussion areas for the performance results of the real-time operation analysis are:

1. The substantial performance benefits achieved between the current EnterpriseOne performance (baseline) compared to an Oracle configuration where the in-memory column store with compression is used and all EnterpriseOne indexes are dropped (with the exception to the primary key integrity constraint index). This is represented by the 'BASLN' and 'BNCHMK InMemory Enabled Dropped Indexes' column values in **Table 1**.
2. The similar results to the above benefit if no EnterpriseOne indexes are dropped, are represented by the 'BASLN' and 'BNCHMK InMemory Enabled' columns.
3. The performance characteristics discussion results from dropping EnterpriseOne indexes and not using the Oracle column store represented by the 'BASLN' and 'BNCHMK InMemory Disabled Dropped Indexes' columns.

Comparison Review: Baseline and Benchmark (InMemory ON, column store enabled)

A detailed analysis of the execution plans of the Oracle optimizer resulted in the following observations. The method the Oracle optimizer chose to achieve the result set is presented in **Table 1**.

1. The Oracle optimizer can find small numbers of records in a large dataset extremely fast when only few columns of information is needed; in the above case, only a single column query was used. For the use cases above, **Table 1** illustrates that when the Oracle in memory column store was enabled, the Oracle optimizer chose an 'INMEMORY SCAN' to achieve its result set.
2. A column store search in-memory is normally much faster than a row based non-indexed column search requiring a full table or index full scan. This was seen in the use cases for columns 'BNCHMK InMemory Enabled Plan' and 'BNCHMK InMemory Enabled Dropped Indexes Plan'. When comparing the results for these two scenarios, the results were very close to being the same. This shows that, for these two sets of use cases, the Oracle optimizer used the in memory column store and that the system performed comparably even with the indexes dropped.
3. When given the choice for a large EnterpriseOne table with a single column query, the Oracle optimizer chose to use the in-memory column store.

As with all use cases, results may vary depending on the data distribution, cardinality, number of rows, and columns queried to name a few of the factors involved. The important message is that dramatic performances are possible with ad-hoc type queries within the EnterpriseOne application when similar conditions exist. **Table 1** demonstrates that this positive outcome was achieved for the ad-hoc query for Sales Order Query (P42101) by a larger order of magnitude of 684.16 times.

Dramatic performance increases can be experienced with real-time operational analysis with the Oracle in memory column store when using non-indexes column queries.

In the Oracle Database In-Memory timed testing where the column store is used, performance results showed orders of magnitude greater for both situations where all of the EnterpriseOne indexes were present and when only the primary key EnterpriseOne index was presented to the Oracle optimizer.

It would be easy to conclude from this that the EnterpriseOne application does not need the performance indexes when implementing the Oracle in-memory column store for those tables. In this limited view of queries, the answer would be in the affirmative. What the reader must realize is that the additional performance indexes have been added to the EnterpriseOne tables to provide performance gains to other EnterpriseOne processes and thus, care must be exercised in removing indexes with the in-memory column store enabled. This is the first message:

Care must be exercised with removing EnterpriseOne indexes for tables placed in the Oracle in-memory column store. Although the primary key index must always remain for integrity reasons, removal of further EnterpriseOne indexes must be followed by rigorous testing of the entire EnterpriseOne modules before implementing in a production environment.

The second message that the results above should highlight is that real-time operational analysis queries ('Ad-Hoc'), or those queries that use non-indexed columns for predicates can benefit greatly in performance when going over large tables. EnterpriseOne standard interactive and batch processes have been developed over time to avoid using these non-indexed columns for performance reasons. The columns are often grayed out in the interactive application and the default batch processes have default data selection to use first the indexed columns of the tables to avoid poor performing processes.

The Oracle in-memory column store provides a mechanism to extend current interactive and batch application queries to include non-indexed columns in their predicate. This can be achieved without the lengthy process of index analysis on poor performing EnterpriseOne processes.

Comparison Review: Baseline and Benchmark (InMemory OFF, column store disabled)

Table 1 also presents the results of tests performed with the Oracle in-memory column store disabled and the EnterpriseOne indexes that were dropped for those tables used primarily in the EnterpriseOne interactive query. The results also showed an improvement to the baseline in all real-time operational analysis queries.

For example, for the Sales Order Query (P42101) application, the timings for the BASLN were 325.661 seconds and for the BNCHMK, the result was 21.670 seconds. However, for the Purchase Order Query (P4310), the values were 31.721 seconds for the BASLN and 29.958 seconds for the BNCHMK test. There are two things about the Oracle optimizer that must be highlighted:

1. The larger number of indexes on a table can cause performance degradation, especially in cases where there are joins between tables. Oracle SQL joins are common in the EnterpriseOne application.
2. The number of current indexes on EnterpriseOne tables has been developed over many years to strike a balance between performance in all of its interactive and batch applications and overhead to the Oracle optimizer in choosing the best plan for execution.

Conclusion

Based on the results of the ad-hoc testing and analysis:

1. The Oracle Database In-Memory has the opportunity to provide significant and dramatic performance improvement when searching over non-indexed database columns.
2. The Oracle Database Optimizer can now evaluate performance decisions regarding existing indexes vs. Oracle Database In-Memory column store options.
3. The database administrator should consider the JD Edwards EnterpriseOne system holistically when considering dropping performance indexes.

Real-time Summarization

EnterpriseOne has many applications that provide summary totals at the end of a query or ‘Find’ operation. When the system returns many thousands of records in this result set, the user must perform a ‘Go to end’ action to navigate to the bottom of the result set and obtain the summary results.

Using the arithmetic and aggregate functionality of the database along with the Oracle 12c columnar format, increased performance improvements can be achieved, as is the case with the EnterpriseOne application P03B2002 - Customer Ledger Inquiry, illustrated by **Figure 5**.

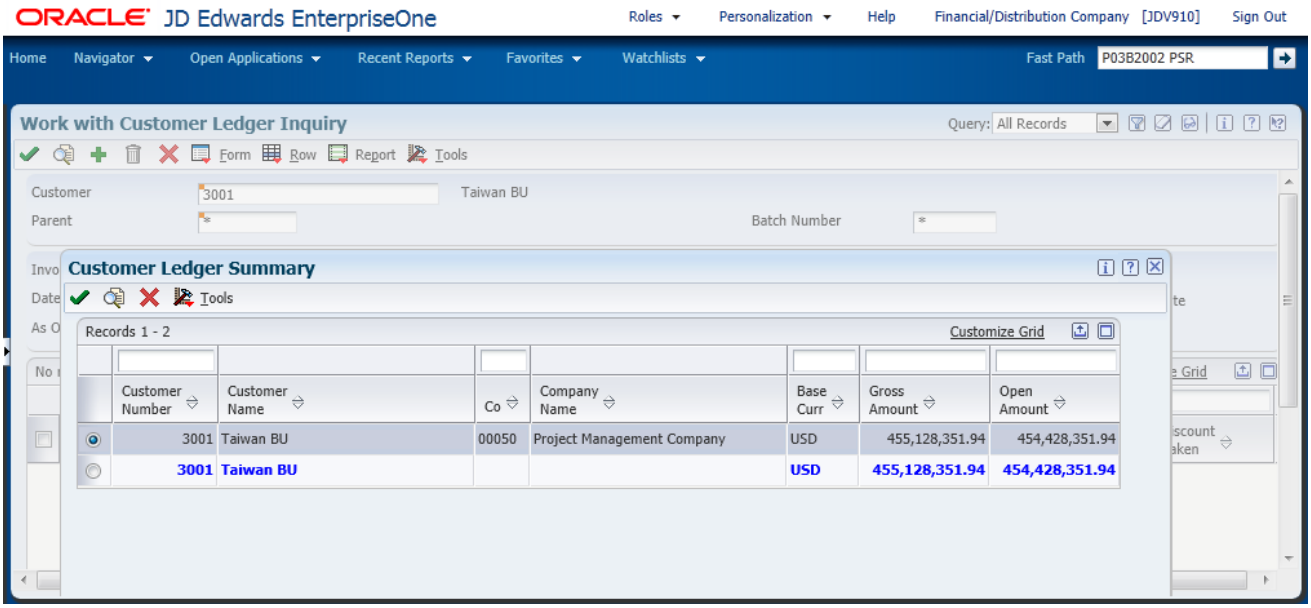

Figure 5: P03B2002 (Customer Ledger Inquiry - Grid Aggregation)

In the Figure 5 example above, the total is presented as a popup result when queried. There is no need to perform a ‘Find’ and then a subsequent ‘Go to end’ function. The summation of the ‘Gross Amount’ is displayed immediately.

This new functionality provides the ability to efficiently view real-time summarized information. For example, month-end books can be closed sooner for Account Ledger applications with the ability to summarize by posted or unposted amounts of currency, or summarize amounts or units by ledger type.

Real-time Summarization Performance Test Results

Table 2 illustrates the result that was achieved for the real-time summarization processes.

The elapsed time was performed and measured using the Oracle software Oracle Application Testing Suite (OATS). The Oracle OATS solution is a comprehensive integrated solution that provides tools to manage the application testing process.

EnterpriseOne Application	Process	BASLN Time (s)	BNCHMK Time (s)	BNCHMK In Memory Enabled Time (s)	BNCHMK In Memory Enabled Dropped Indexes Time(s)	X-Times Faster (rounded)
Supplier Ledger Inquiry	**P0411**	4.986	0.262	0.813	0.216	**23.1**
Work with Forecasts	**P3460**	1.438	0.346	0.325	1.733	**4.4***
Purchase Order Inquiry	**P4310**	239.645	38.403	0.625	1.185	**202.2**
Account Ledger	**P09200**	21.075	0.434	0.475	1.553	**13.6**
Account Ledger by Object Account	**P09201**	92.846	0.439	0.589	1.184	**78.4**
Account Ledger by Category	**P09202**	83.318	27.778	21.513	6.175	**13.5**
Customer Service Inquiry	**P42101**	412.258	60.514	26.022	7.207	**57.2**
Customer Ledger Inquiry	**P03B2002**	4952.560	1.998	3.227	1.654	**2994.3**

Table 2: EnterpriseOne Real-time Summarization

Note: * The **c**alculation of the Work with Forecast process is based on the comparison between the 'Baseline' column and the 'Benchmark with In-Memory Enabled' column whereas the other processes are based on the comparison between the 'Baseline' column and the 'Benchmark with In Memory Enabled Dropped Indexes' column.

The results presented in **Table 2** are to show the combined benefits in performance of both the code enhancement and implementing the Oracle 12c In-Memory feature. The following terms are defined here to assist in the discussion of the results:

Process: The real-time summarization process is the EnterpriseOne application designation that was used in testing. There are two versions of this real-time summarization application used in testing. The first EnterpriseOne application is the original process that is currently available; the second is an enhanced version of the process available January 2015.

Time(s): The time value was the server-side time required to perform an action in the EnterpriseOne application process. For the Oracle 12c baseline example, time represents the server-side timings to perform the 'Find' and the 'Go to end' actions. In the case of the Oracle 12c benchmark testing, the action to bring up the summarization report was reported. The summarization report performs an internal 'Find' operation and therefore needs to be compared to the composite 'Find' and 'Go to end' action timings measured in the baseline testing. The timings represent an 'average' of server-side response times over 10 iterations of the actions for each EnterpriseOne application.

As a final comment, in all of the baselines cases, only the server-side timing of the action was measured. In a normal EnterpriseOne environment, in order to obtain the summary totals that are presented in the benchmark testing as a normal course of the interactive user experience, the user using the original code would perform the 'Find', 'Go to end', and then a series of manual steps to export the result set into an Excel spreadsheet and perform the summary calculations in that application.

Oracle 12c Baseline: The baseline is defined as the PRE-code feature enhancement (original code) of the EnterpriseOne application that was used in testing. The testing of the original code was performed on Oracle Exadata Database Machine X3-2 with Oracle Database 12.1.0.2, but without the Oracle Database In-Memory feature enabled.

Oracle 12c Benchmark: The Oracle 12c Benchmark column is the result of the testing of the enhanced EnterpriseOne features, also on Exadata X3-2 and Oracle Database 12.1.0.2 without Database In-Memory.

Oracle 12c Benchmark with In-Memory Enabled: This column represents testing where both row and column format representations of selected EnterpriseOne tables are enabled using the Oracle Database In-Memory feature.

Oracle 12c Benchmark with In-Memory Enabled and Dropped Indexes: This column represents the results of testing selected EnterpriseOne tables using the Oracle Database In-Memory feature and dropping the EnterpriseOne non-integrity related indexes. For more information on EnterpriseOne indexes and the Oracle Database In-Memory feature see:

White Paper: *Oracle Database In-Memory with Oracle's JD Edwards EnterpriseOne*
https://apex.oracle.com/pls/apex/f?p=44785:141:109388643419968::::P141_PAGE_ID%2CP141_SECTION_ID:121%2C1344

Real-time Summarization Analysis – Baseline and Benchmark Comparisons

Figures 6, 7, and 8 below illustrates the large performance gains resulting from both the EnterpriseOne application enhancements and performance gains achieved by placing selected EnterpriseOne tables into the Oracle Database In-Memory column store. The comparisons will be between the columns 'Baseline', 'Benchmark with In-Memory Enabled', and 'Benchmark with In-Memory Enabled and Dropped Indexes'.

The "X-Times Faster" performance gain was measured by dividing the 'Baseline' by the 'Benchmark with In-Memory Enabled and Dropped Indexes' columns. For example, in the first row of Table 2, a (23.1 value (4.986/0.216) represents 2208% faster ((4.986-0.216)/0.216) between the values of baseline and benchmark for P0411 (Supplier Ledger Inquiry).

The timing values for each test are presented above each graph bar for direct purposes of comparison. They have been rounded to the first significant decimal from the actual value presented in **Table 3**.

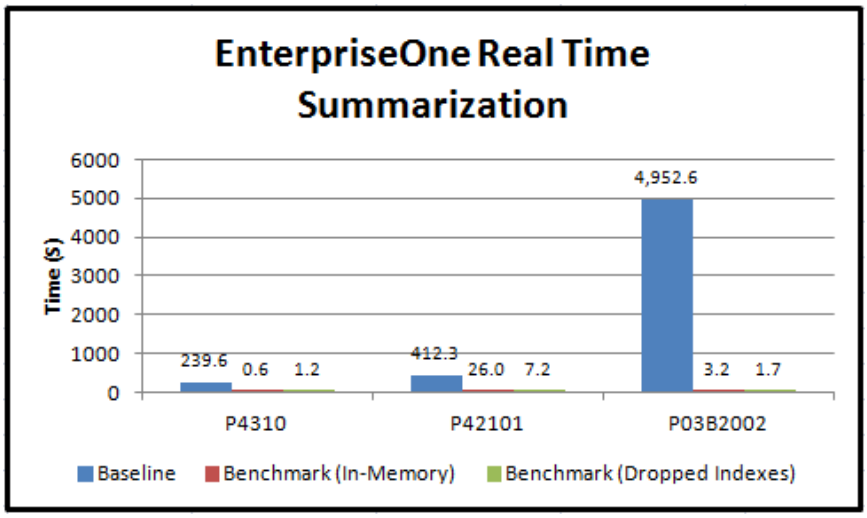

Figure 6: P4310, P42101, and P03B2002 Analysis

In **Figure 6**, we illustrate the performance gains of three of the major applications; namely P4310 (Purchase Order Inquiry), P42101 (Customer Service Inquiry), and P03B2002 (Customer Ledger Inquiry). Performance improvements of 202.2x, 57.2x, and an enormous gain of 2994.3x were observed during the testing. Performance increases for P4310 were seen for the benchmark testing for the 'In-Memory' and 'In-Memory with Dropped Indexes' tests. These results will be discussed in more detail in the next section. Otherwise, good improvements to overall performance were found.

Note: Baseline timings do NOT include the time for the manual steps required to export the results of the 'Find' and 'Go to end' actions to an Excel spreadsheet and then perform the summary calculations manually. In all benchmark testing and with the new enhanced code, the summarizations are presented to the user dynamically.

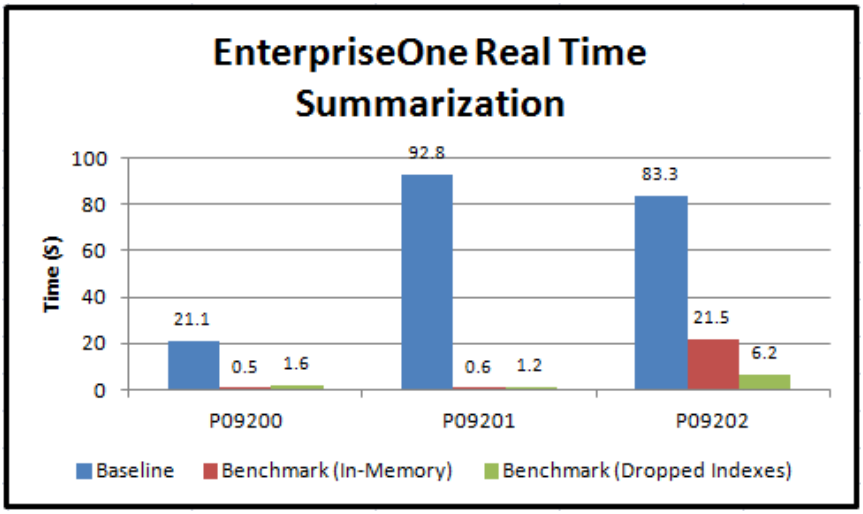

Figure 7: P09200, P09201, and P09202 Analysis

Figure 7 illustrates the results of the Account Ledger EnterpriseOne applications, namely P09200 (Account Ledger), P09201 (Account Ledger by Object Account), and P09202 (Account Ledger by Category), similar performance increases of 13.6x, 78.4x, and 13.5x were observed. Performance increases for P09200 and P09201 were seen for the benchmark testing for the 'In-Memory' and 'In-Memory with Dropped Indexes' tests. These results will be discussed in more detail in the next section.

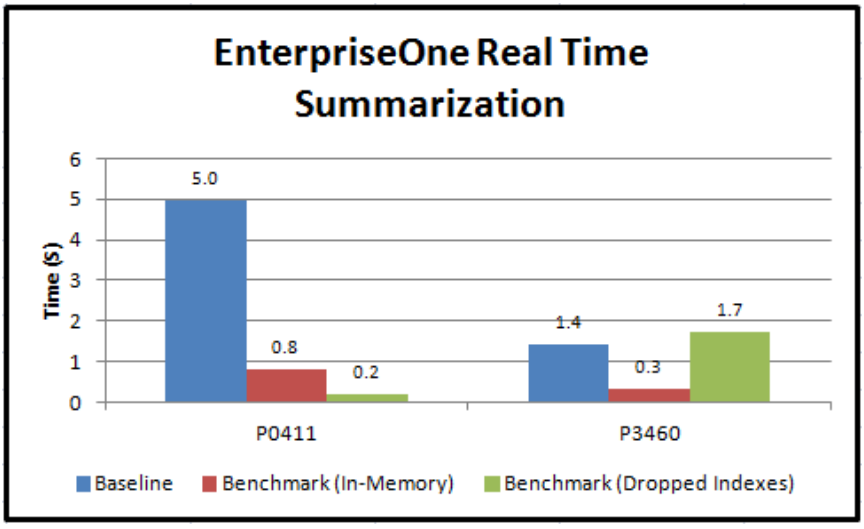

Figure 8: P0411 and P3460 Analysis

Figure 8 illustrates the last two real-time summarization EnterpriseOne applications; P0411 (Supplier Ledger Inquiry) and P3460 (Work with Forecasts). As stated in the last two analyses of results, performance aspects of P3460 will be discussed in the next section.

Positive performance improvements were seen in all tests conducted between the Baseline and Benchmark EnterpriseOne applications. Performance concerns between the different configurations of the Oracle In-Memory database will be the discussion of the next section.

Analysis - Comparisons with JD Edwards EnterpriseOne Enhancements and Database In-Memory Feature

In **Figures 9 and 10** the analysis will concentrate on looking at the performance differences between the EnterpriseOne applications where the EnterpriseOne code remains the same. In this case, the EnterpriseOne enhanced code is kept constant and testing was performed to look at the benefits of only the Oracle Database In-Memory column store.

The timing values for each test are presented above each graph bar for direct purposes of comparison. They have been rounded to the first significant decimal from the actual value presented in Table 2.

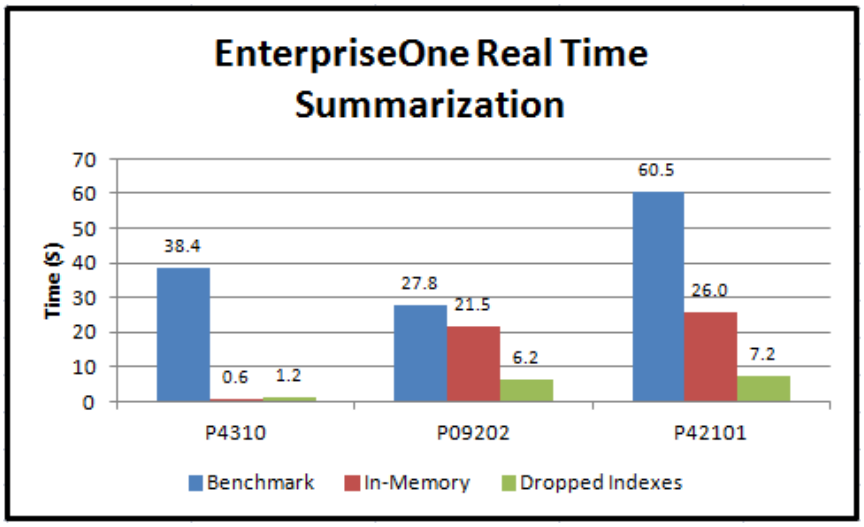

Figure 9: Column store analysis of P4310, P09202, and P42101

Figure 9 shows the performance gains of the three EnterpriseOne applications; P4310 (Purchase Order Inquiry), P09202 (Account Ledger by Category, and P42101 (Customer Service Inquiry). Performance improvements of 32.4x, 4.5x, and 8.4x are directly related to placing selected EnterpriseOne tables into the column store and dropping non-integrity indexes.

For the case of P4310, although there is an increase to performance for the Benchmark compared to 'In-Memory ON' and 'In-Memory with Dropped Indexes', there is NOT an decrease as is seen in P09202 and P42101 between 'In-Memory ON' and 'In-Memory with Dropped Indexes'. Values of 0.6 and 1.2 seconds were seen respectively for the P4310 EnterpriseOne application. The following section on 'Explaining Performance Variations' will attempt to explain some of these values.

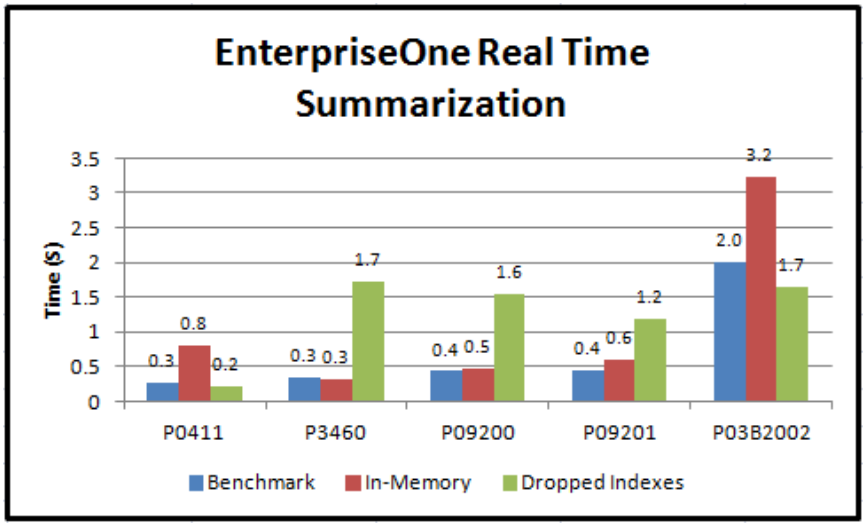

Figure 10: Column store analysis of P0411, P3460, P09200, P09201, and P03B2002

Figure 10 illustrates that performance gains are not always found when selected EnterpriseOne table are placed in the column store. In this set of tests, P0411 (Supplier Ledger Inquiry), P3460 (Work with Forecasts), P09200 (Account Ledger), P09201 (Account Ledger by Object Account), and P03B2002 (Customer Ledger Inquiry) showed mixed results. The detail analysis of these results will be the topic of the next discussion point in this document.

Real-time Summarization Analysis - Explaining Performance Variations

To fully understand the variables involved in understanding WHY these results are found in testing, a view of the number of records processed and the size of the number of records in the main tables that are queried needs to be discussed. It must be noted that the difference in the values in all of the tests in Figure 10 are relatively small (i.e. a few seconds). It is in this observation that the discussion of the results will be centered on.

Table 3 below provides further information about each process and the makeup of the data including the number of records processed by the application and the total number of records in the main table used by the EnterpriseOne application.

EnterpriseOne Application	Process	Primary Table Queried	Records Processed	Total Records Primary Table	Recommend In Memory	Recommend In Memory Dropping Indexes
Supplier Ledger Inquiry	**P0411**	**F0411**	**4,437**	**34,759**	**NO**	**YES**
Work with Forecasts	**P3460**	**F3460**	**1,852**	**152,210**	**YES**	**NO**
Purchase Order Inquiry	**P4310**	**F4311**	**178,559**	**5,151,544**	**YES**	**YES**
Account Ledger	**P09200**	**F0911**	**35,420**	**9,581,669**	**YES**	**NO**
Account Ledger by Object Account	**P09201**	**F0911**	**35,865**	**9,581,669**	**YES**	**NO**
Account Ledger by Category	**P09202**	**F0911**	**35,139**	**9,581,669**	**YES**	**YES**
Customer Service Inquiry	**P42101**	**F4201**	**8,800**	**15,398,285**	**YES**	**YES**
Customer Ledger Inquiry	**P03B2002**	**F0911**	**400,000**	**9,581,669**	**NO**	**YES**

Table 3: EnterpriseOne Real-time Summarization Table Information

Performance differences in Figure 10 revolve around HOW the Oracle optimizer is choosing the best plan to execute the SQL that EnterpriseOne is submitting. Because row and column formats of the data are available to the Oracle optimizer, it has to make a decision as to which format it will be using.

Two of the key components of this calculation:

- *The number of records that will be processed in the query*
- *The key tables in the query and the total number of records that must be searched in row and column formats*

In the simplest terms, upon detailed analysis of each of the tests using AWR reports and analyzing many execution plans, it was determined that the differences in the timings was mainly due to the additional overhead that the Oracle optimizer needed to choose the best plan for executing the SQL request. Large numbers of records processed as compared to the total number of records in the table was one factor. In the example of P0411, 4,437 records were processed where the main table had only 34,759 total records. The original benchmark timing was 0.262 seconds (0.3 seconds rounded up for presentation purposes in Figure 10), 0.813 seconds for testing with Oracle 12c with In-Memory turned ON, and 0.216 seconds when Oracle 12c In-Memory column store was used and all non-integrity indexes for selected EnterpriseOne tables were dropped. Similar analysis was performed for the P4310 EnterpriseOne application resulting in the same conclusion.

Conclusion

Based on the results of the testing and analysis, **Table 3** provides what would be the recommended configuration based on the current data set and testing results for real-time summarization. The recommendation is for implementing the enhanced code for the given EnterpriseOne applications. This recommendation is simplistic, because it is only based on the limited testing that was performed in this analysis. For production environments, a more rigorous evaluation would be implemented.

Using **Table 3**, performance gains in P4310, P09202, and P42101 would result if either 'In-Memory' or 'In-Memory with Dropped Indexes' were implemented. Performance gains for P0411 and P03B2002 would be found only with 'In-Memory with Dropped Indexes'. Finally, P3460, P09200, and P09201 would be the most efficient in an 'In-Memory' configuration.

As observed through this discussion, an EnterpriseOne OLTP application of real-time summarizations can have mixed results when implementing the Oracle 'In-Memory' feature. What can be said is that using the In-Memory feature will provide a performance benefit to the application. What is the caveat is the level of indexes that can be dropped to achieve peak performance.

Last Note: The performance impact due to the maintenance of table indexes is well documented for insert, update, and delete operations. Real-time summarization EnterpriseOne applications do not demonstrate this performance impact adequately. Real-time summarization applications did not perform a significant amount of these actions during testing and thus should be continually evaluated.

For detailed instructions on running these new real-time summarization applications, see:

JD Edwards EnterpriseOne Applications Accounts Payable Implementation Guide
http://docs.oracle.com/cd/E16582_01/doc.91/e15084/process_ap_vouchers.htm#sthref306

JD Edwards EnterpriseOne Applications Accounts Receivable Implementation Guide
http://docs.oracle.com/cd/E16582_01/doc.91/e15085/workcustomerledgerinfo.htm#sthref754

JD Edwards EnterpriseOne Applications General Accounting Implementation Guide
http://docs.oracle.com/cd/E16582_01/doc.91/e15112/reviewbalandtransaction.htm#sthref892
http://docs.oracle.com/cd/E16582_01/doc.91/e15112/reviewbalandtransaction.htm#sthref900
http://docs.oracle.com/cd/E16582_01/doc.91/e15112/reviewbalandtransaction.htm#sthref908

JD Edwards EnterpriseOne Applications Forecast Management Implementation Guide
http://docs.oracle.com/cd/E16582_01/doc.91/e15111/work_w_so_history.htm#sthref275

JD Edwards EnterpriseOne Applications Procurement Management Implementation Guide
http://docs.oracle.com/cd/E16582_01/doc.91/e15131/enter_purchase_orders.htm#sthref471

JD Edwards EnterpriseOne Applications Sales Order Management Implementation Guide
http://docs.oracle.com/cd/E16582_01/doc.91/e15146/enter_sales_orders_for_csrs.htm#sthref750

Real-time Financial Reconciliation

The EnterpriseOne real-time financial reconciliation in the financials module was the target of product enhancements. These traditional batch reports, converted to interactive programs, used the techniques of aggregation, arithmetic's across columns, and set based operations providing the user to view integrities in seconds rather than hours.

One example of a reconciliation report is the UBE batch report R007032 (Batch out of Balance) written as the interactive program P007032 as illustrated in **Figure 11**. Each of the real-time financial reconciliation programs provides interactive capabilities such as exits and hyperlinks to the direct batch numbers of information responsible for the out of balance condition so that they could be directly accessed and corrected in real time.

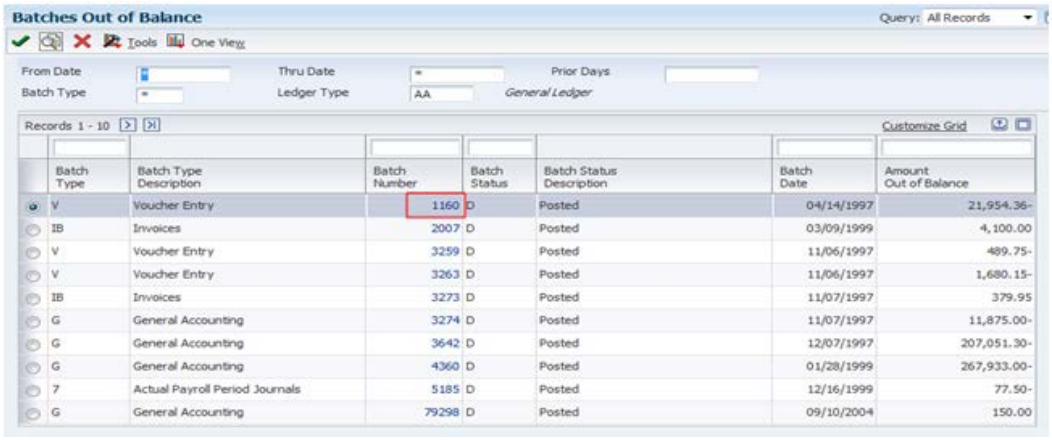

Figure 11: P007032 (Interactive Batch out of Balance Reconciliation Report)

This new functionality provides an interactive, real time user interface to correct issues which is a dramatic contrast to the batch process that just reported the out of balance conditions in a PDF report. In the traditional batch processing of this report, additional manual steps were then required to correct the integrity issues by reviewing the results of the PDF, and then going into the application to find where the out of balanced conditions were and to reconcile them with the PDF report.

The batch process was then rerun to validate that the reconciliation was complete. In the interactive real time process, after reconciling in real time the out of balance condition, the information is automatically updated providing a current view in real time of the integrity of a company's financials. Each interactive program has similar real time capabilities that gives users one program to review and correct issues. The results of the real-time financial reconciliation conversion processes and the time savings, as compared to their traditional batch process, are presented in the Results section of this document.

Real-time financial reconciliation provided by the combination of the Database In-Memory feature and EnterpriseOne product enhancements can dramatically change the end user experience, thus transforming how a business processes and approaches its financial reconciliation process.

Real-time Financial Reconciliation Performance Test Results

Table 4 illustrates the result that was achieved for the real-time financial reconciliation processes, also by significant orders of magnitude. The change of the financial integrities from batch to an interactive process would not be possible if the time of execution was not improved from hours to seconds.

The elapsed time was performed and measured using the Oracle software Oracle Application Testing Suite (OATS). The Oracle OATS solution is a comprehensive integrated solution that provides tools to manage the application testing process.

EnterpriseOne Application	Baseline Process	Elapsed Time (s)	Benchmark Process	Elapsed Time (s)	X-Times Factor
Batch to Detail	**R007021 and R007031**	10,305	**P0072131**	99.538	**104x**
Batch out of Balance	**R007032**	7,331	**P007032**	69.546	**105x**
Company Out of Balance	**R097001**	1,928	**P097001**	38.403	**50x**
Transaction to Account Master	**R097021**	9,807	**P097021**	51.502	**190x**
Account Balance to Account Master	**R097031**	4,685	**P097031**	17.210	**272x**
Accounts to Business Unit	**R097041**	488	**P097041**	5.883	**83x**
Company by Batch Out of Balance	**R09706**	7,369	**P09706**	62.998	**117x**

Table 4: EnterpriseOne Real-time Financial Reconciliation

For detailed instructions on running these integrity reports as well as the new real-time financial reconciliation applications, see:

JD Edwards EnterpriseOne Applications General Accounting Implementation Guide
http://docs.oracle.com/cd/E16582_01/doc.91/e15112/verifydataintegrity_sf.htm#EOAGA01041

Conclusion

Performance characterization of selected JD Edwards EnterpriseOne applications with the Database In-Memory feature yielded the following results:

1. The ad-hoc query use case demonstrated a *684* times improvement over the same query running against data that was not in memory. Based on these results, customers are encouraged to discover their own use cases for real-time operational analytics.
2. The real-time summarizations demonstrated between *2 and 2994* times better when comparing performance gains between the original EnterpriseOne code (Baseline) and EnterpriseOne application code that was enhance and where selected EnterpriseOne tables were placed In-Memory (with Dropped Indexes). When comparing the performance of the enhanced EnterpriseOne code and gains due to only the Oracle In-Memory performance, a *4 to 32* times performance gains was observed.
3. In general, the real-time financial reconciliation reports achieved a significant order of magnitude performance improvement, ranging from *50 to 272* times faster than the former processes.
4. The financial reconciliation and summarization processes was further simplified by the creation of hyperlinks and other interactive features to these interactive applications that allowed for a more streamlined end-user experience.

Enabled by the power of Oracle Database In-Memory, JD Edwards EnterpriseOne customers have a low-risk, low-impact path to in-memory computing, allowing them to transform their business processes and gain the insight, efficiency, and competitiveness of a real-time enterprise.

See Also:

- Web site: Oracle Database In-Memory
  http://www.oracle.com/us/products/database/options/database-in-memory/overview/index.html
- ://www.oracle.com/technetwork/database/in-memory/overview/twp-oracle-database-in-memory-2245633.html
- White Paper: Oracle Database In-Memory (October 2014)
  http://www.oracle.com/technetwork/database/in-memory/overview/twp-oracle-database-in-memory-2245633.html?ssSourceSiteId=ocomen
- Oracle Database Documentation Library: https://docs.oracle.com/database/121/nav/portal_4.htm
- White Paper: Oracle Database In-Memory with Oracle's JD Edwards EnterpriseOne
  https://apex.oracle.com/pls/apex/f?p=44785:141:109388643419968::::P141_PAGE_ID%2CP141_SECTION_ID:121%2C1344

Appendix A: Environment Details

EnterpriseOne Architecture Components

EnterpriseOne Web Server

Hardware

- Exalogic X3-2 VM Base Configuration
- 32 x Intel Xeon E5-2690 2.9 GHz processors
- 128 GB of memory

Software

- Oracle WebLogic Server 11g (10.3.5)
- Exalogic Enhanced enabled

EnterpriseOne Logic Server

Hardware

- Exalogic X3-2 VM Base Configuration
- 32 x Intel Xeon E5-2690 2.9 GHz processors
- 128 GB of memory

Software

- EnterpriseOne Application Software 9.1 Update 2
- EnterpriseOne Tools 9.1.4.7 for Ad-Hoc
- EnterpriseOne Tools 9.1.5 for Real Time Financial Reconciliation and Real Time Summarization

EnterpriseOne Oracle Database Server

Hardware

- Exadata X3-2 2 Node RAC configuration
- 32x Intel Xeon E5-2690 2.9 GHz processors per node
- 512 GB of memory per node

Software

- Oracle Database 12c (12.1.0.2)
- 2 Node RAC configuration
- In-Memory Column Store enabled
- In-Memory default compression level ‘MEMCOMPRESS FOR QUERY LOW’. Optimized for query performance.

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Hardware and Software, **Engineered to Work Together**

Performance Characterization of Oracle JD Edwards EnterpriseOne with the Oracle Database In-Memory
March 2015