Banner Enterprise Data Warehouse

Baseline ETL Improvements

July 2013

In preparing and providing this publication, Ellucian is not rendering legal, accounting, or other similar professional services. Ellucian makes no claims that an institution's use of this publication or the software for which it is provided will guarantee compliance with applicable federal or state laws, rules, or regulations. Each organization should seek legal, accounting and other similar professional services from competent providers of the organization's own choosing.

Prepared by: Ellucian
4375 Fair Lakes Court
Fairfax, Virginia 22033
United States of America

Revision History Log

Publication Date	Summary
July 2013	New version that supports Banner Enterprise Data Warehouse (EDW) 8.3, 8.4, 8.4.1, 8.4.2, and 8.4.3 software.

Table of Contents

Banner EDW ETL improvements

EDW patch p1-1d7fk0z includes improvements to the EDW ETL (extract, transform, and load) processes in the following two areas: performance and error handling.

Performance improvements

The performance improvements greatly improve the overall timing of Load and Refresh processes. These improvements include:

- Support for parallel processing as well as new parallel “Load All” and “Refresh All” ETL jobs.
- Improvements for bitmapped index creation (using parallelization and asynchronous jobs).
- Updated driver cursor logic for some extract table functions.
- Converting the get_student extract process from a table function to a procedure and adding internal parallelization.

Error handling improvements

The error handling improvements make the ETL processes more fault tolerant when they encounter unexpected data relationships. These improvements consist primarily of controller logic that runs the “WKEYS” mappings, then checks for constraint violations and handles those data rows with issues. This allows the rest of the WKEYS data to flow in to the FACT table/mapping. The rows with errors are noted in the job Control Report and also stored in an exceptions table that you can use to help you evaluate and correct the issues.

BPRA product patch requirements

This EDW patch requires corresponding patches for related Banner Performance Reporting and Analytics (BRPA) products. Refer to the following table to identify which patches you should apply for your configuration of BPRA products.

BPRA product	Patch
ODS 8.3	p1-1dvf1jl_ods8030052
EDW 8.3	p1-1d7fk0z_edw8030017
EDW 8.4	p1-1d7fk0z_edw8040031
EDW 8.4.1	p1-1d7fk0z_edw8040124
RAP 1.2.1	p1-1fez4fh_rap1020102
SRP 1.0	p1-1fez4fj_srp1000001
ODS 8.4	p1-1dvf1jl_ods8040012
EDW 8.4.2	p1-1d7fk0z_edw8040214
RAP 1.2.2	p1-1fez4fh_rap1020201
SRP 1.0.1	p1-1fez4fj_srp1000101
ODS 8.4.1	p1-1dvf1jl_ods8040109
EDW 8.4.3	p1-1d7fk0z_edw8040307
RAP 1.3	p1-1fez4fh_rap1030001
SRP 1.1	p1-1fez4fj_srp1010001

Parallel ETL processing

The EDW Operational Star ETL jobs consist of multiple sets of OWB mappings that run in sequence. For each star, there is a Load and a Refresh job (where the Refresh job only processes changes since the last Refresh). Each job includes OWB mappings that run in a specific sequence as follows:

INPUT → CLEAN → DIM(S) → WKEYS → FACT

where each of the above represents a mapping (or group of mappings) that must run in sequence (the INPUT loads the data for the CLEAN, the CLEAN loads data for the DIM(S), and so on).

Each star job includes multiple DIM (Dimension) mappings which run in sequence. However, each DIM mapping reads from the CLEAN table (populated by the CLEAN mapping) and writes to a different Dimension table, so the DIM mappings can be run in parallel if there are multiple processors to accommodate them.

For example, if a star has ten DIM mappings and they each take ten minutes to run, they will take a total of 100 minutes to run in sequence. When run in parallel (assuming you have at least eleven processors) the entire set can complete in a little more than ten minutes instead of 100.

Running the mappings concurrently is not only more efficient due to simultaneous processing but also because it can use the Oracle database caching feature. All of the mappings are reading from the same table, so repeated requests for the same data can be filled from the cache instead of reading from the disk storage.

By reorganizing the groupings of mappings though, they can be run in parallel. The following two configurations are delivered to optimize EDW ETL "Load/Refresh All" processing:

- Narrow – Runs standard sequence with DIMs in parallel
- Wide – Runs all INPUT/CLEAN pairs in parallel, then DIMs for each star in parallel (sequentially by star), and then all WKEYS/FACTs in parallel.

Standard Administrative UI job definitions for both Narrow and Wide are delivered and run using the Administrative UI. The updated EDW Load jobs are delivered in the Administrative UI in the installed Process called "**Schedule Banner EDW Operational Mappings – Parallel"**. All of the load jobs run in the Narrow configuration, while a "Load/Refresh All (WIDE)" job is also available.

Parallel bitmapped index processing

The Fact table in each EDW star uses bitmapped indexes (BMIs) to link to its related dimension tables for fast reporting access. Bitmapped indexes are a standard data warehouse optimization to improve reporting. The down side is that BMIs are slow and expensive to update. Each time the FACT mapping runs, it first drops the BMI and loads the Fact table, and then recreates the BMI afterward. BMI creation is sequential and for stars with large amounts of data (and many BMIs), this process can take significant time. However, running each of the BMI create processes in a separate job; can also achieve notable overall performance improvement. The baseline EDW ETL processes have been updated to perform all BMI creation in parallel jobs.

In almost all cases the BMIs are not needed on the stars until after the ETL processing is complete, so creation of these can be deferred to the end of the processing cycle. This allows for other database activities/tasks to run alongside the BMI creation tasks.

Dynamic extract logic optimization

For selected stars (Student and Student Course), the extract package logic has been updated to improve performance. The first optimization (for both stars) was to convert the extract driver cursor SQL from a static cursor to a dynamic one. Within these driver cursors, the same SQL is used for both Load and Refresh modes, with an optional subquery used for Refresh that references the Change table to identify only rows changed since the last refresh. Normally this subquery does not significantly impact the execution plan for the SQL. However, with more complex SQL logic, the optional subquery can have performance implications. Dynamically generating the SQL at run-time (based on L vs. R) allows the Oracle SQL compiler to determine the best execution plan (depending if the Change table is used or not). The Student and Student Course extract packages have been updated to use this feature.

Parallel extract logic

For complex extract logic, internal parallelization of the driver cursor can improve overall performance by splitting a large result set into multiple individual jobs. This technique was applied to the Student star. As a result, the Student star will submit multiple jobs when run. Each job processes a subset of data (loading it into the CLEAN table) and the calling/parent job waits for them all to complete before returning. To control (“throttle”) the number of jobs, you can configure a database parameter (MTVPARM) to limit the maximum number of “child” jobs for a single execution of the extract logic. What parameter is this? A new parm I assume?

CLEAN Table Stats Gathering Consolidated

Each of the CLEAN mappings was set to gather Oracle statistics on the CLEAN table to improve the execution plans for the subsequent DIM and WKEYS mappings that access it. However, the Cleansing process (called as a post-mapping procedure from the mapping) was also gathering table statistics. In addition, the call to gather statistics from each OWB mapping was not set up to take advantage of Parallel Query Server (PQS). To eliminate this duplicate work, the CLEAN mappings have all been updated to disable statistics gathering, leaving the Cleansing process to perform the DBMS_STATS call consistently. Also, the statistics call from Cleansing Process uses the Degree parameter to enable PQS, which greatly improves the performance of the statistics queries.

Improved WKEYS Error Handling Logic

The WKEYS mappings merge the data in the CLEAN table with the dimensions to populate the dimensional surrogate IDs for each row being loaded into the fact table. As part of that processing, the mapping validates the granularity of the data loaded in the fact table, where the WKEYS table constraints will restrict any ‘bad’ data. In the past, this meant that any ‘bad’ data would cause the WKEYS mapping to stop the entire ETL operation for that star.

This process has been improved to better handle any data exceptions, identify them in the Control Report, and then move those errant rows of data to an exception table that you can review and correct later. The rest of the ‘good’ data then flows into the WKEYS table (and in to the star). Refer to the document “Error_Handling_Feature.pdf” for additional information about how to diagnose and correct WKEYS errors. This document is available from the Support Center Documentation Download area under the Banner EDW product.

Parallel ETL environment configuration considerations

Ensuring proper database/server configuration is critical to achieving the performance gains described in this document because nearly all of these improvements involve parallelization. These enhancements are implemented using the standard Oracle DBMS_JOBS infrastructure and so are controlled using the standard Oracle initialization parameter job_queue_processes. This parameter defines the maximum number of concurrent jobs. You will want to maximize both processor and memory resources. You will need to do some investigation to determine the best setting for the job_queue_processes parameter. It is recommended that you start by setting the value to 2 less than the number of processor cores on the database server. Similarly, the database server must have sufficient memory to allow it to run an ETL job in each of those job queues. We recommend a minimum of 1 GB per job thread.

Example configurations:

- If the DB server has 32 cores and 32 Gigs of memory, set the job_queue_processes = 30.
- If the DB server has 16 cores and 16 Gigs of memory, set the job_queue_processes = 14.
- If the DB server has 1 core, setting the job_queue_processes to 3 will basically run the jobs in close to a serial fashion (i.e., the “parent” job will take 1 process and then each child processes will run sequentially in the other.)

> **Note:** Never set the job_queue_processes lower than 3 because that will cause the parent job to “hang” as it waits for the submitted child jobs to run (the EDW ETL Load/Refresh _All jobs have a 3 layer dependency in some areas).

Oracle will adjust the job scheduling environment based on a number of factors besides job_queue_processes (including resource plans, available resources, and so on) so these recommendations do not supersede Oracle’s guidelines. Refer to the Oracle Database Server Administration documentation for more information.

Disk Space

Disk space is another consideration of ETL processing. As each Star load runs, it populates various temporary (temp) tables (INPUT, CLEAN, and WKEYS tables) as it moves the data through the process. When run in the “Narrow” mode, the data in all three temp tables is purged when the star ETL completes. This means that the maximum disk space needed is only enough space for the largest star to run. When run in “Wide” mode, however, all of the CLEAN tables will be populated at the same point in time because all of the INPUT/CLEAN pairs run together. This means that you may need a significant amount of free space can for the jobs to complete. (In one case, .25 TB). After all the FACT mappings have completed, that space will be cleaned up, but for that intermediary time (while the DIMs/WKEYS are running) the space is needed while all of the CLEAN tables are populated.

The Control Report for the “Load/Refresh All” jobs now lists the amount of space used in the temp tables as each star runs. It is recommended to first run the Load job in “Narrow” mode, and then determine the amount of space needed to run in “Wide” mode by summing up the listings from the Control Report. A sample SQL query is provided (estimate_wide_space.sql) to help with this.

Disk space should not be an issue for Refresh processing because there are typically a smaller number of rows changing in a Refresh job (provided you run the refresh on a frequent/nightly basis). For this reason, running the Refresh job in “WIDE” mode should provide the best refresh performance while not requiring too much disk space.

Administrative UI improvements

The Administrative UI has been updated to support running parallel parent/child job threads. Each child job generates its own Control Report (CR). When a child job finishes, that Control Report data is moved to the parent Control Report. This consolidates CR information into one report and removes the need for multiple child Control Reports. In addition, any errors/warnings are reflected in the parent job status.

Because each child job is a standard job when running, you can manage and stop it using the standard Administrative UI processes. The Control Report infrastructure has been updated to display “child” records differently so that you can track concurrent job execution timings while still supporting the standard CSV format export.

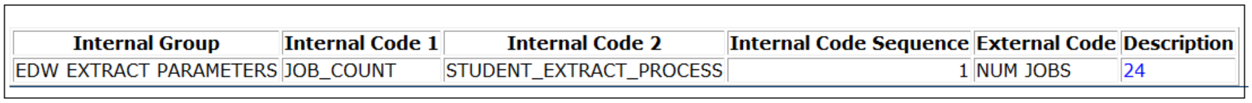
The “Set Up A Parameter” pages have been updated with additional links to provide easier navigation for the various parameters (ETL CONTROL GROUP, ETL MAP PACKAGE, and ETL PACKAGE) related to a job.

EDW Extract Parameters

There are some new EDW Extract Parameters that you can use to customize aspects of the parallel processing feature.

JOB_COUNT EDW Extract Parameter

The JOB_COUNT EDW Extract Parameter controls the number of concurrent “child” jobs run by the ETL process for the Student star. Change the value in the Description field for this parameter to the number of “child” jobs that you want the ETL process to run concurrently. You should typically set this value to match the job_queue_processes value. Refer to the “Parallel extract logic” for more information.

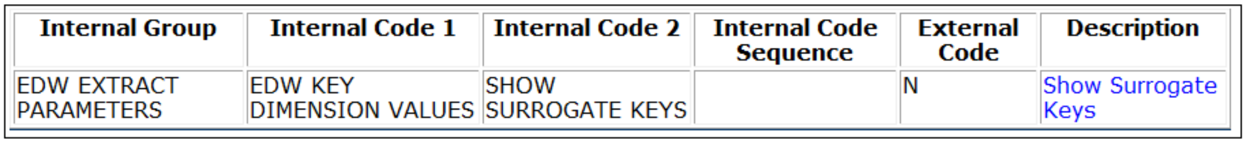


Internal Group	Internal Code 1	Internal Code 2	Internal Code Sequence	External Code	Description
EDW EXTRACT PARAMETERS	JOB_COUNT	STUDENT_EXTRACT_PROCESS	1	NUM JOBS	24

EDW KEY DIMENSION VALUES EDW Extract Parameter

The value in the External Code field of the EDW KEY DIMENSION VALUES EDW Extract Parameter controls whether the WKEYS surrogate key values are displayed in the control report in addition to the cleansed key values from the dimension tables. This is primarily used for debugging and not needed in most cases. Set the External Code value as follows:

- Y = Show surrogate key values
- N = Do not show surrogate key values



Internal Group	Internal Code 1	Internal Code 2	Internal Code Sequence	External Code	Description
EDW EXTRACT PARAMETERS	EDW KEY DIMENSION VALUES	SHOW SURROGATE KEYS		N	Show Surrogate Keys

RETAIN_ETL_TEMP_DATA EDW Extract Parameter

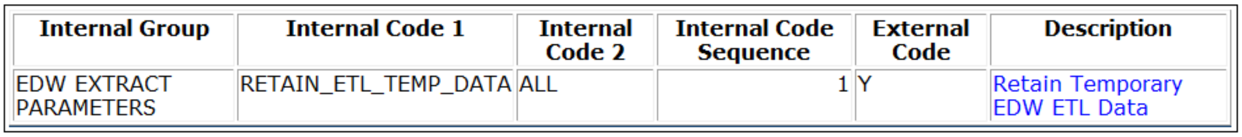
Use the RETAIN_ETL_TEMP_DATA EDW Extract Parameter to retain the temporary ETL data (in the INPUT, CLEAN, and WKEYS tables) for a given star load. Normally this data is deleted after the FACT_INSERT mapping runs. However, you can set this Extract Parameter to ‘Y’ to indicate that the mapping retain the data. You can then use the data for evaluation/debugging purposes.

You can set up this logic for ALL stars or for an individual star. No records for this EDW Extract Parameter are delivered with the product so you must create records as follows.

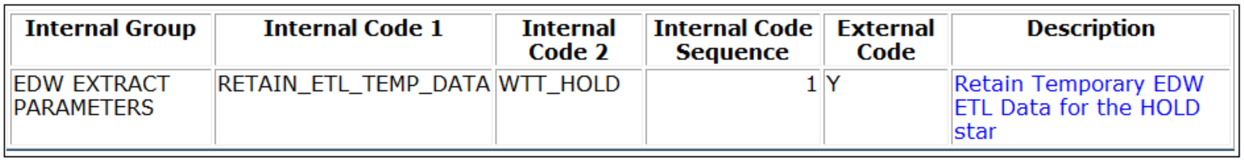
- Internal Group = “EDW EXTRACT PARAMETERS”

- Internal Code 1 = “RETAIN_ETL_TEMP_DATA”
- Internal Code 2 = “ALL” to apply the logic to all stars or the name of the Star Temp tables to apply the logic to selected stars, for example, “WTT_HOLD” or “WTT_STUDENT”
- External Code = Y
- Description = a description of the rule logic, for example, “Retain Temporary EDW ETL Data”

See the following examples of this EDW Extract Parameter.



Internal Group	Internal Code 1	Internal Code 2	Internal Code Sequence	External Code	Description
EDW EXTRACT PARAMETERS	RETAIN_ETL_TEMP_DATA	ALL	1	Y	Retain Temporary EDW ETL Data



Internal Group	Internal Code 1	Internal Code 2	Internal Code Sequence	External Code	Description
EDW EXTRACT PARAMETERS	RETAIN_ETL_TEMP_DATA	WTT_HOLD	1	Y	Retain Temporary EDW ETL Data for the HOLD star

Note: The data in these tables is still “temp” data; however, so subsequent runs of the ETL processes will replace the contents of these tables. This means that the data is only kept until the next ETL cycle. If you need to keep the data for a longer period of time, consider backing up this data in another way as well.

RETAIN_ETL_CHG_DATA EDW Extract Parameter

This EDW Extract Parameter is similar to the RETAIN_ETL_TEMP_DATA parameter. Use the RETAIN_ETL_CHG_DATA parameter to keep CHG table data for a given star or for ALL stars. This parameter is only intended to support debugging efforts because CHG records should get cleared out each time you run a Refresh job. (If they are kept for a longer period of time, they will start to impact performance.)

Set this Extract Parameter to ‘Y’ to indicate that the mapping keep the change data. You can set up this logic for ALL stars or for an individual star. No records for this EDW Extract Parameter are delivered with the product so you must create records as follows.

- Internal Group = “EDW EXTRACT PARAMETERS”
- Internal Code 1 = “RETAIN_ETL_CHG_DATA”
- Internal Code 2 = “ALL” to apply the logic to all stars or the name of the Star Change tables to apply the logic to selected stars, for example, “WTT_HOLD” or “WTT_STUDENT”

- External Code = Y
- Description = a description of the rule logic, for example, “Retain Change EDW ETL Data”