Enhancing ETL Performance with Warehouse Builder

Manole VELICANU, Larisa COPCEA (TEOHARI)
University of Economic Studies, Bucharest, Romania
mvelicanu@yahoo.com; larisa.copcea@yahoo.com

We live in a dynamic world, in a permanent move, were performance of information systems continually increases. Thus, the need of being informed, regardless of place or time, is very great using data warehouses solutions. Therefore, the need for efficient information systems is very high. The efficiency can be achieved by using various methods/techniques and technologies "to build" the data warehouse. The ones, we will present in our paper, are: methods of Enhancing ETL Performance with Warehouse Builder: the purpose of ETL strategies is to create an integrated, complex, coherent software solution; techniques of data warehouse optimization: in order to improve the performance of data warehouse processing can be applied several optimization techniques; and "in terms" of technology, we will consider: data warehouses (using Oracle Warehouse Builder).

Keywords: *performance, Oracle Warehouse Builder, efficiency of information systems, extraction, transformation, and loading (ETL), data warehouse optimization*

1 Introduction

Here we focus on what we can do through Warehouse Builder, that is, performance configuration setting during ETL and schema design. Tips on configuring your database for optimized performance, we focus mainly on specific Warehouse Builder performance-related features to achieve optimal performance.
Consider involving your organization's data architect in the very first planning steps, well before Warehouse Builder is implemented. It is too late to start thinking about performance strategies at the point where you are using Warehouse Builder to create ETL mappings.
As the data warehouse size is increasing and crossing terabytes limits, and as the query turnaround time is getting shorter, administrators have the additional overhead of monitoring the performance of the data warehouse system regularly. A major task of any data warehouse system is ETL, and it is essential that the ETL design be tuned for optimized performance. There are multiple places in a Warehouse Builder implementation where an administrator can look for possible performance bottlenecks. These are: hardware, operating system, network, database, application (that is, Warehouse Builder).

2. ETL Design: Mappings

Extraction, transformation, and loading in a production data warehouse can be very time consuming and take up a lot of your system resources. Yet, when implementing a warehouse, the focus is more on how to have a perfect dimensional model rather than how to create ETL logic with run-time performance in mind. It is important to know that a well designed ETL mapping can make all the difference in the performance of your data warehouse. Consider the Cost of Function Calls.
For example, a simple replacement of a function operator with a join condition operator in the mapping can make huge difference in the time taken by the mapping

to execute. This is because if you are including a function in a mapping, the function call will take the context out to the function and after the function completes, the control returns back into the mapping. So there is extra time taken. Or when joining from multiple tables, it is better to stage the results in a staging table and then apply a filter or a aggregate operator than doing it all on the fly.

Hence, it is a good ETL design decision to keep the context switches to a minimum and make use of the various mapping operators that Warehouse Builder provides to accomplish different tasks.

You can review your mappings on some of the performance parameters. You need to answer questions such as: Which operating mode is better for the ETL logic you are using, or what will be the best commit strategy or will parallel DML enhance performance, and so on. If you are moving large volumes of data from remote source, will it be better to do an import/export or to use transportable modules. The below *figure 1* shows a mapping created in Warehouse Builder to load the dimension customers.

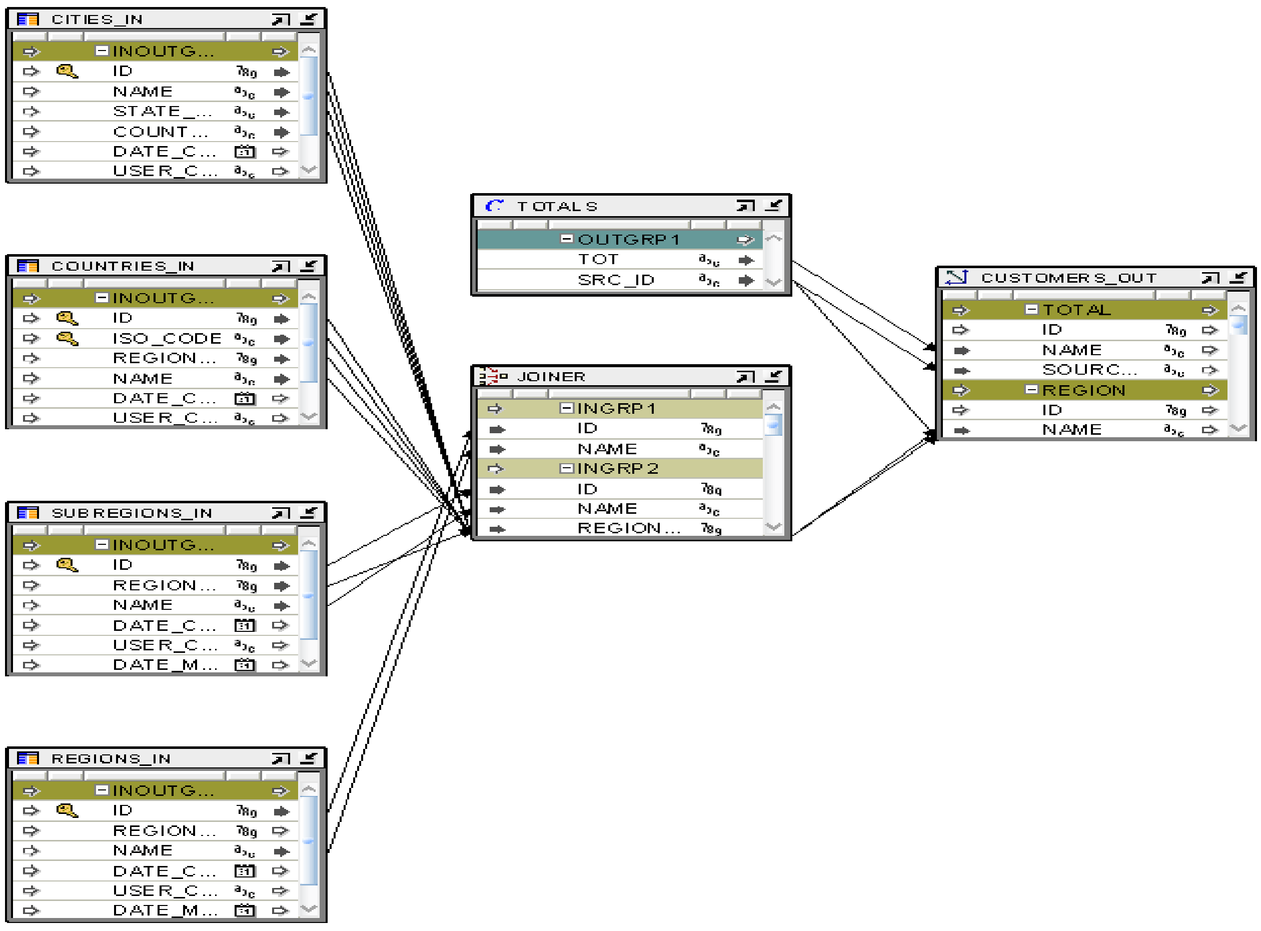
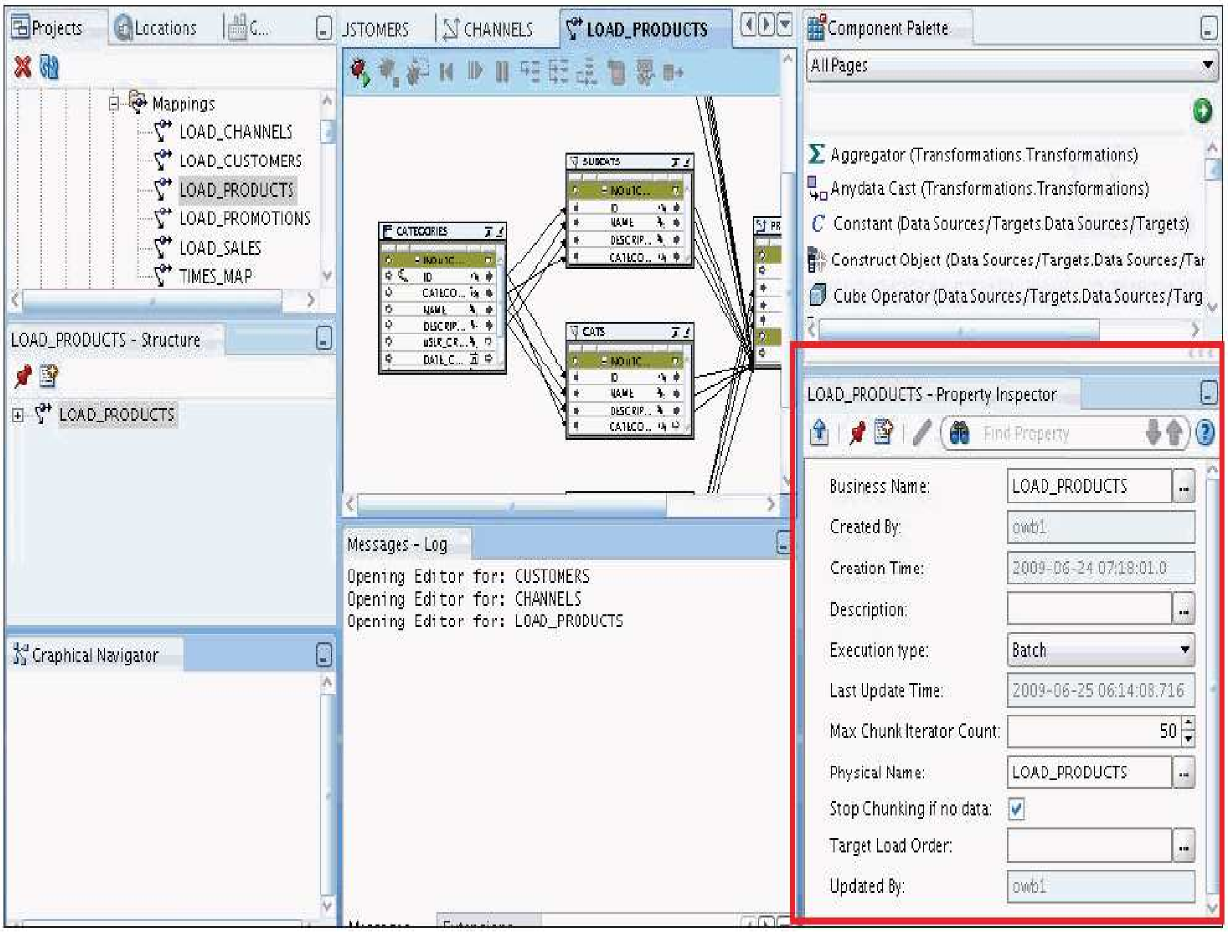


Fig 1. Oracle Warehouse Builder mapping load dimension customers
Source: author


Projects
Locations
Mappings
LOAD_CHANNELS
LOAD_CUSTOMERS
LOAD_PRODUCTS
LOAD_PROMOTIONS
LOAD_SALES
TIMES_MAP
LOAD_PRODUCTS - Structure
LOAD_PRODUCTS
Graphical Navigator
JSTOMERS
CHANNELS
LOAD_PRODUCTS
CATEGORIES
SUBCATS
CATS
Component Palette
All Pages
Aggregator (Transformations.Transformations)
Anydata Cast (Transformations.Transformations)
Constant (Data Sources/Targets.Data Sources/Targets)
Construct Object (Data Sources/Targets.Data Sources/Tar
Cube Operator (Data Sources/Targets.Data Sources/Targ
LOAD_PRODUCTS - Property Inspector
Find Property
Messages - Log

```
Opening Editor for: CUSTOMERS
Opening Editor for: CHANNELS
Opening Editor for: LOAD_PRODUCTS
```

Business Name:	LOAD_PRODUCTS
Created By:	owb1
Creation Time:	2009-06-24 07:18:01.0
Description:	
Execution type:	Batch
Last Update Time:	2009-06-25 06:14:08.716
Max Chunk Iterator Count:	50
Physical Name:	LOAD_PRODUCTS
Stop Chunking if no data:	✓
Target Load Order:	
Updated By:	owb1

Fig. 2 Oracle Warehouse Builder design center for mappings
Source: author

The above *figure 2* shows the design center for mappings in Warehouse Builder.
The "Set based" and "Row based" operating modes have their advantages and disadvantages. The "Set based" operating mode is the fastest way to process data. The rows are processed as a single dataset, with a single SQL statement that is either completed successfully for all the rows or fails if there is a problem with even one row. You cannot view details about which rows contain errors.
The "Row based" operating mode gives more control to the user, both in terms of logging row-by- row processing information, as well as in terms of control over the management of incorrect data. Warehouse Builder generates statements that process data row by row. The records can be extracted from the source one at a time, processed, and inserted in the target. If there is a problem with some rows, this is logged and the processing continues. Detailed logging of row-by-row processing information is available.
The default operating mode you select depends upon the performance you expect, the amount of auditing data you require, and how you design the mapping.
The following are the five operating modes, ranked by performance speed, with the fastest first: "Set based", "Set based fail over to row based (target only)", "Set based fail over to row based" (the default), "Row

based (target only)" and "Row based" The default operating mode is "Set based fail over to row based," in which mode Warehouse Builder will attempt to use the better-performing "Set based" operating mode, but will fall back to thes lower "Row based" mode if data errors are encountered. This mode allows the user to get the speed of "Set based" processing, but when an unexpected error occurs it allows you to log these errors. The "Row based (target only)" and "Set based fail over to row based (target only)" operating modes are a compromise between the "Set based" and the "Row based" modes. The "target only" modes will use a cursor to rapidly extract all the data from the source, but will then use the "Row based" mode to insert data into the target, where errors are more likely to occur. These modes should be used if there is a need to use the fast "Set based" mode to extract and transform the data as well as a need to extensively monitor the data errors.
Warehouse Builder generates code for the specified default operating mode as well as for the unselected modes. Therefore, at run time, you can select to run in the default operating mode or any one of the other valid operating modes. The types of operators in the mapping may limit the operating modes you can select. As a general rule, mappings run in "Set based" mode can include any of the operators except for Match-Merge, Name-Address, and Transformations used as procedures [4].

3. Commit Control and Audit Level within Warehouse Builder

By default, Automatic is selected for the Commit Control run-time parameter. You may use the automatic commit when the consequences of multiple targets being loaded unequally are not great or are irrelevant because a mapping has multiple targets. Warehouse Builder commits and rolls back each target separately and independently of other targets. For PL/SQL mappings you can override the default setting and control when and how Warehouse Builder commits data by using either Automatic Correlated or Manual. Automatic Correlated: If you want to populate multiple targets based on a common source, you may also want to ensure that every row from the source impacts all affected targets uniformly. When Automatic Correlated is selected, Warehouse Builder considers all targets collectively and commits or rolls back data uniformly across all targets. Correlated commit operates transparently with PL/SQL bulk processing code. The correlated commit strategy is not available for mappings run in any mode that is configured for Partition Exchange Loading or includes an Advanced Queue, Match-Merge, or Table Function operator.
The role of the Commit frequency run-time parameter is to enable the user to decide how often data is committed. You should select a number that will not put too much strain on the rollback segments size. The default is set to 1,000 rows. Commit frequently!
Use "Default audit level" to indicate the audit level used when executing the package. Audit levels dictate the amount of audit information captured in the run-time schema when the package is run. You can set it to NONE, ERROR DETAILS, STATISTICS, or COMPLETE. The default audit level will define how detailed the audit information collected during the load process will be. Running a mapping with the audit level set to Complete generates a large amount of diagnostic data, which may quickly fill the allocated tablespace and can impact performance.
When you select STATISTICS, statistical auditing information is recorded at run time.

4. Additional Run-Time Parameters for Mappings

You can configure additional run-time parameters for your mappings to tune performance as discussed below:

A. "Bulk size" and "Bulk processing code": If "Bulk processing code" is set to True, the "Row based" mode will process the rows in bulks instead of individual rows, which will improve the performance of the row-based mappings. In this case, the bulk size will be given by the "Bulk size" run-time parameter. It is recommended to keep the bulk size value around 50, which is the optimal value.

B. "Maximum number of errors": Use "Maximum number of errors" to indicate the maximum number of errors allowed when executing the package. Execution of the package terminates when the number of errors reached is greater than the "Maximum number of errors" value. The maximum number of errors is only relevant in the Row based operating mode. This parameter will set the limit of data errors that will be tolerated during the mapping run, before the mapping is stopped for too many errors. If the mapping is stopped due to this, the mapping ends in ERROR.

5. Partition Exchange Loading (PEL) in Warehouse Builder

You can use Partition Exchange Loading (PEL) to load new data by exchanging it into a target table as a partition. This exchange process is a DDL operation with no actual data movement. PEL is patented technology specific to Warehouse Builder. Only Oracle has this technology. One major advantage is the ability to do parallel direct path loading. Before Oracle9i, if a table was partitioned to multiple partitions, the server could only serialize a load to one partition at a time. To solve that problem, PEL technology was created for OWB, allowing a non-partitioned staging table to hold the data. Another main advantage of PEL is the ability to load data while not locking the target table. The swapping of names and identities requires no time. The target table is not being touched. One minor advantage of using PEL is to avoid rebuilding the indexes and constraints in the big partitioned table—data is loaded, indexes created, and constraints maintained in the staging table (on the much smaller scale), and after the loading is completed, the staging table is rotated and replaces a big table partition in a single stroke [3].

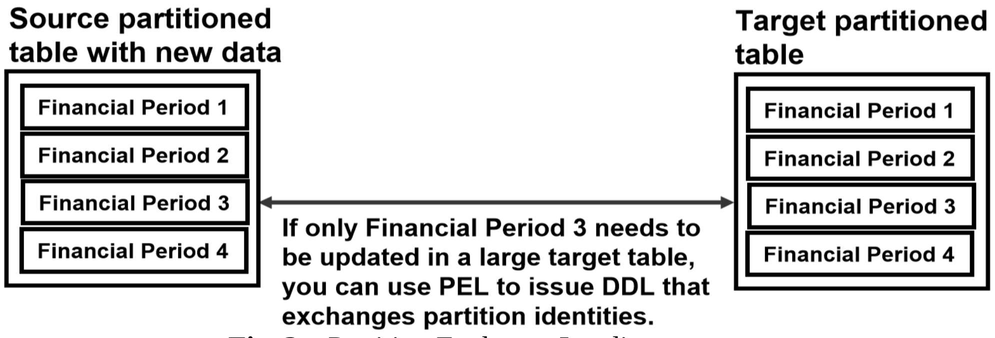

Source partitioned table with new data
Financial Period 1
Financial Period 2
Financial Period 3
Financial Period 4
Target partitioned table
Financial Period 1
Financial Period 2
Financial Period 3
Financial Period 4
If only Financial Period 3 needs to be updated in a large target table, you can use PEL to issue DDL that exchanges partition identities.

Fig. 3 – Partition Exchange Loading
Source: [4]

In this example from *figure 3*, only one financial period needs to be updated in a large target table with much historical data. Instead of issuing a SQL Delete command, PEL uses data definition language (DDL) statements to swap partition assignments, without the data movement required of data manipulation language (DML) statements.

Guidelines to achieve PEL for targets:
• The large table must be partitioned by range, based on a single date field.
• Partition names must respect a specific naming convention.
• The staging table must contain data for the same date range as the partition it is going to replace.
• Data and indexes must each be located in a single tablespace. (All the partitions and the staging table must be in a single “data” tablespace, while the indexes can be located on a single “index” tablespace.)
• The structure of the staging table must be identical to the structure of the large partitioned table, and they have to have the same indexes and constraints.
• All the indexes in the big partitioned table must be local indexes (pertaining to a single partition).

There are two options for PEL: Direct and Indirect.
Direct PEL: The user designs and maintains the staging table that is switched into the large partitioned table directly. This usually happens in a mapping that has a one-to-one correspondence between the source (the staging table) and the target (the large partitioned table). Direct PEL is convenient because it allows the user to physically separate the process of loading (when the staging table is loaded) from the process of publishing the data (when the data is swapped into the large partitioned table and made available to the query/reporting users).
For example, the data loading into the staging table can be scheduled during the day, while the actual publishing of the data, which might be disruptive to the query/reporting users (front-end users), can be scheduled during the night when there is little or no front-end activity.
Another way to use direct PEL (which switches old data from the partition to the staging table) is to manage “dormant” data (data that is not often used by the analytical users, but needs to be available on short notice). In this case, empty staging tables can be swapped with the least-used partition data (usually the oldest data) making the oldest partition empty, but maintaining the actual data in the staging table. The data can then be switched online almost instantaneously by re-running the mapping.

Indirect PEL: Warehouse Builder creates a temporary staging table behind the scenes, swaps the data with the partition, and deletes the table once the swap is completed. This kind of configuration will be necessary when the mapping loads the data into a target table from remote sources or by joining multiple source tables [4].

6. Using Transportable Modules for Data Extraction from Remote Sources

A very common occurrence in data loading is for the target processes to access data in remote sources. A problem here might be caused by the database link. Since a mapping is a single session, the database link will create a single communication channel between source and target, which will force the data to travel sequentially, thus creating a bottleneck in the data movement process.
The solution here is to use transportable modules. Using transportable modules, you can copy large volumes of data and metadata, from an Oracle 9i database to another Oracle 9i database, and from an Oracle 10g database to another Oracle 10g database or 11g database. However, you cannot use transportable modules to copy from an Oracle 9i to an Oracle 10g database. If both versions are 10g, you can create transportable modules to copy data and metadata between Oracle databases on

different machine architectures and operating systems. For 10g and 11g databases, you specify either Data Pump or transportable tablespaces during configuration of the transportable module. In the case of 10g databases and Oracle Data Pump, you can transport tables without also transporting their tablespaces. For example, if your table is 100 KB and its tablespace size is 10 MB, you can deploy the table without deploying the entire tablespace. Furthermore, only Data Pump gives you the option to copy the entire schema.

Benefits of Using Transportable Modules
Previously, to transport data you relied on moving flat files containing raw data. These mechanisms required that data be unloaded or exported into files from the source database, and then these files were loaded or imported into the target database. Transportable modules entirely bypass the unload and reload steps and gives you access to the Transportable Tablespaces and Data Pump Oracle server technologies. The following are the benefits of using a transportable module [3]:
• *High performance data extraction*: Transportable modules reduce the need for Warehouse Builder mappings to access data remotely. If you have large volumes of data on remote machines, use transportable modules to quickly replicate the sources onto the Oracle target database.
• *Distribute and archive data marts*: Normally a central data warehouse handles ETL processing while dependent data marts are read-only. You can use transportable modules to copy from a read-only data mart to multiple departmental databases. In this way, you can use your central data warehouse to periodically publish new data marts and then replace old data marts simply by dropping the old tablespace and importing a new one.
• *Archive sources*: You can set your source tablespaces to read-only mode and then export them to a target. All the data files are copied creating a consistent snapshot of the source database at a given time. This copy can then be archived. The advantage of this method is that archived data is restorable both in the source and target databases.

7. Best Practices Tips: Factors That Impact Performance

The following are a few simple ETL design practices that influence the performance of your mappings considerably:
• *Custom transformation impact*: Transformation functions should be used sparingly on large tables. The reason is that the generated SQL statements containing the transformation function call will force the database engine to switch between the SQL engine (that interprets pure SQL statements) and PL/SQL engine (that interprets the procedural structures, such as functions). Whenever possible it is beneficial to replace simple PL/SQL functions with pure SQL expressions.
• *Loading type impact*: This will determine which SQL statement will be used to update the target. It is worth noting that the INSERT statement is the least inefficient of the operations available, because it creates new data only, so the user should use INSERT whenever possible. Insert/Update and Update/Insert are slightly more inefficient (these load types will generate a MERGE statement when the mapping is configured as "Set based"). If the user expects most of the operations to be inserts, the user should choose the INSERT/UPDATE loading type and vice versa. Pure UPDATE and DELETE are inefficient and should be used sparingly.

• *External table vs. SQL Loader*: Warehouse Builder still retains the possibility of using SQL Loader to rapidly load flat files into the database. When deciding whether to use external tables or SQL Loader mappings, the user should find the right trade-off between the conveniences of the external table (that can be used in any mapping operation as a normal read-only table) and the necessity to load the flat file data as fast as possible into a real database table. This might be a better solution for very large flat files, because external tables have a number of limitations (they cannot be indexed, keys cannot be created, etc.), which can make them inefficient when included in mappings with more complex operators (joins, filters, etc.).

Warehouse Builder provides design capabilities for indexes, partitions, and allowing for detailed configuration of physical storage and sizing properties on objects.

7.1 Indexing

Indexes are important for speeding queries by quickly accessing data processed in a warehouse. You can create indexes on one or more columns of a table to speed SQL statement execution on that table. You can create UNIQUE, B-tree, Bitmap (non-unique), Function-based, Composite, and Reverse indexes in Warehouse Builder. Bitmap indexes are primarily used for data warehousing applications to enable the querying of large amounts of data. These indexes use bitmaps as key values instead of a list of row IDs.

Bitmap indexes can only be defined as local indexes to facilitate the best performance for querying large amounts of data. Bitmaps enable star query transformations, which are cost-based query transformations aimed at efficiently executing star queries. A prerequisite of the star transformation is that a bitmap index must be built on each of the foreign key columns of the cube or cubes. When you define a bitmap index in Warehouse Builder, set its scope to LOCAL and partitioning to NONE. Local indexes are likely to be the preferred choice for data warehousing applications due to ease in managing partitions and the ability to parallelize query operations.

Another widely used performance enhancement method is dropping the indexes of the target object before the loading process and recreating the indexes after the load is completed. This can significantly improve the performance, because indexes will not have to be maintained during the load.

You need indexes only when using reporting tools to query the loaded data.

There is no switch or check box option that can enable you to switch the indexes on or off as required. To achieve this, the user will have to create a pre-mapping that will invoke a PL/SQL function or procedure that drops the target indexes and a post-mapping process that will invoke PL/SQL code that will recreate these indexes.

7.2 Constraints Management

Constraints management can also dramatically affect performance of "Set based" mappings. If the Enable Constraints property is checked, Warehouse Builder will leave the target object constraints (foreign keys or any other constraint) enabled during the load. This might make the load dramatically slower because the constraints will have to be checked against every row that is loaded into the target. If this property is unchecked, on the other hand, the target foreign key constraints will be disabled before the beginning of the load and re-enabled (in parallel) after the load is completed. This will make the load faster, but if rows that do not conform to the constraints are loaded, the affected rows in the target object will be marked as invalid during the constraint re-activation. The user will then have to manually correct these rows whose row IDs will be logged in the run-time audit error table or, if specified, in an exceptions table. For "Row based" mappings, the constraints will be active no matter what the setting is for this parameter.

7.3 Partitions

Partitions enable you to efficiently manage very large tables and indexes by dividing them into smaller, more manageable parts. Use partitions to enhance data access and improve overall application performance, especially for applications that access tables and indexes with millions of rows and many gigabytes of data [2]. Partitioning greatly enhances the manageability of the partitioning table by making the backups, restores, archives, etc. much easier to perform.

Partitioning can also enhance the performance by giving the user more control on how to optimally configure the physical parameters of the table. If there is a single large table, it will physically reside in a single Oracle tablespace and probably in a single data file. The user will not have control over where this file is physically located on a disk or array of disks. If a parallel operation is performed on this table and the table resides on a single disk or mostly on a single disk, the disk and its controller will represent a bottleneck for any activity on this table. What use is it having several parallel processes trying to load data into the table if they all have to write or read sequentially through a single disk controller?

You can define various types of partitions in Warehouse Builder. Range partitioning is the most common type of partitioning and is often used to partition data based on date ranges. For example, you can partition sales data into monthly partitions. When you design mappings using a table that is range partitioned on a date column, consider enabling Partition Exchange Loading (PEL), already discussed earlier.

Specifying Partition Tablespace Parameters

If the table is partitioned, the user can assign every partition to a different tablespace and every tablespace to a different disk, spreading the data evenly across disks. This will make it possible for the server processes to balance the processing activity among themselves evenly, thus making the parallel execution much more efficient. If you neglect to specify partition tablespaces, Warehouse Builder uses the default tablespaces associated with the table and the performance advantage for defining partitions is not realized. You need to specify this when the partition type is one of the following: HASH BY QUANTITY, RANGE-LIST, RANGE-HASH, or RANGE-HASH BY QUANTITY. You can also specify the Overflow tablespace list.

7.4 Parallelism

The Warehouse Builder run-time environment has been designed for parallel processing. This implies that users running the tool on a platform with more than two processors will benefit the most from parallel processing. If the object is configured as parallel, Warehouse Builder will make sure that when the object is deployed, it will be created for parallelism by adding the parallel option (which is checked by the database engine when a statement is run against the object) to the CREATE statement (during the first table deployment CREATE TABLE … PARALLEL… will be executed). For any query executed against this table, the database engine will attempt to launch multiple parallel processes to enhance query performance.

Although this can significantly enhance query performance for the reporting users, the downside of this might be the possible extensive use of resources (especially memory) that parallel queries require in a multi-user, high-query-volume environment.

With objects enabled for parallel access, you need to set the Enable Parallel DML option for the mappings to take full advantage of parallelism. For target objects in the mappings, Warehouse Builder by default adds the PARALLEL hint.

If you enable Parallel DML, Warehouse builder will always generate the ALTER SESSION ENABLE PARALLEL DML statement in a PL/SQL mapping. The implication is that Warehouse Builder will always attempt to execute the mapping in parallel if the objects involved are enabled for parallelism.

• If you have only one CPU, do not use Parallel DML. It will lower performance by launching multiple processes that share the same CPU, requiring process context switches that involve huge overhead.

• If you have two CPUs, Parallel DML might be useful.

• If you have a dual-core CPU, Oracle does not yet have a recommendation, as the implications are currently being studied.

You can set tablespace properties (for indexes as well as objects) at various levels:

• *User level*: If no tablespace is specified, the objects go into the tablespace assigned to the user you are deploying to. When you create your target users, you have the option of specifying the tablespaces to be used.

• *Module level*: To allow overriding specification of the tablespaces (index and object) you can set this at module level, all generated objects will take this property unless overridden at the object level. In the Configuration Properties window for the specific module, you can set the Default Index Tablespaces and Default Object Tablespaces property.

• *Object level*: Allows specific control per object for both indexes and objects. In the Configuration Properties window for the specific object, you can set the Tablespace property.

8. Conclusion

The complexity of the information systems used in a company has grown along with its expansion and increase in its volume of sales or along with the increase in their number of employees. So, the gathered information is useful only if it is of dependable quality and is delivered at the right time. In the same time, the need of software integration comes as a must for data that need to be integrated and the creation of complex, robust, efficient and finally, complete software solutions for data warehouses [1]. Configure your ETL and schema design performance parameters and use Warehouse Builder to create and configure indexes, partitions, and constrain are important matters to be applied for the data warehouse system performance.

References

[1] Manole Velicanu, Daniela Liţan, Larisa Copcea (Teohari), Mihai Teohari, Aura-Mihaela Mocanu (Vîrgolici), Iulia Surugiu, Ovidiu Răduţă, Ways to increase the efficiency of information systems, The 10th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Databases (AIKED ’11), University of Cambridge, UK, 2011, vol Recent Researches in Artificial Intelligence, Knowledge Engineering and DataBases, pp. 211-216, ISSN: 1792-8117/1792-8125, ISBN: 978-960-474-273-8, (ISI Thomson).

[2] N. T. Nguyen, M. T. Le, J. Swiatek, Intelligent Information and Database Systems, Springer-Verlag, Berlin Heidelberg, 2010.

[3] B. Griesemer, Oracle Warehouse Builder 11g Getting Started, Packt Publishing, 2009.

[4] www.otn.oracle.com

Manole VELICANU is a Professor at the Economic Informatics Department at the Faculty of Cybernetics, Statistics and Economic Informatics from the Academy of Economic Studies of Bucharest. He has graduated the Faculty of Economic Cybernetics in 1976, holds a PhD diploma in Economics from 1994 and starting with 2002 he is a PhD coordinator in the field of Economic Informatics. He is the author of 22 books in the domain of economic informatics, 64 published articles (among which 2 articles ISI indexed), 55 scientific papers published in conferences proceedings (among which 5 papers ISI indexed and 7 included in international databases) and 36 scientific papers presented at conferences, but unpublished. He participated (as director or as team member) in more than 40 research projects that have been financed from national research programs. He is a member of INFOREC professional association, a CNCSIS expert evaluator and a MCT expert evaluator for the program *Cercetare de Excelenta - CEEX* (from 2006). From 2005 he is co-manager of the master program *Databases for Business Support*. His fields of interest include: Databases, Design of Economic Information Systems, Database Management Systems, Artificial Intelligence, Programming languages.

Larisa COPCEA (TEOHARI) has graduated the Academy of Economic Studies (Bucharest, Romania), Faculty of Cybernetics, Statistics and Economic Informatics in 2006. She holds a Master diploma in Databases - Support for business from 2008 and in present she is a Ph.D. Candidate in Economic Informatics with the Doctor's Degree Thesis: Advanced management of information in data warehouses.
Her research activity can be observed in the following achievements: 6 proceedings (papers ISI proceedings), among witch:

- "Ways to Increase the Efficiency of Information Systems", Proc. of the 10th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Databases (AIKED '11, University of Cambridge), February 20-22, 2011, Cambridge, UK;
- „Some Information Technologies to Improve the Performance of an ERP System", Proc. of the 5th WSEAS International Conference on Computer Engineering and Applications (CEA '11), January 29-31, 2011, Puerto Morelos, Mexico;
- "XML Authoring Tool", Proc. of the 4th European Computing Conference (ECC'10,University Politehnica of Bucharest), April 20-22, 2010, Bucharest, Romania;

and 3 articles published in scientific reviews , among witch:

- "Technologies for Development of the Information Systems: from ERP to e-Government", International Journal of Applied Mathematics and Informatics, issue 2, vol. 5, 2011.

Her scientific fields of interest include: Data warehouse, Databases, Database Management Systems, High availability solutions, Information Systems and Economics.