IMPROVING QUERY PERFORMANCE IN VIRTUAL DATA WAREHOUSES

ADELA BÂRA
ION LUNGU
MANOLE VELICANU
VLAD DIACONIȚA
IULIANA BOTHA
Economic Informatics Department
Academy of Economic Studies
Bucharest
ROMANIA
ion.lungu@ie.ase.ro
manole.velicanu@ie.ase.ro
bara.adela@ie.ase.ro
diaconita.vlad@csie.ase.ro
iuliana.botha@ie.ase.ro

Abstract: In order to improve the quality of Business Intelligence Systems in an organization we can choose to build the system using BI techniques such as OLAP and data warehousing or by using traditional reports based on SQL queries. The cost and developing time for BI tools is greater than those for SQL Reports and these factors are important in taking decisions on what type of techniques we used for BIS, also the problem of low performance in data extraction from data warehouse can be critical because of the major impact in the using the data from data warehouse: if a BI report is taking a lot of time to run or the data displayed are no longer available for taking critical decisions, the project can be compromised. In this case there are several techniques that can be applied to reduce queries' execution time and to improve the performance of the BI analyses and reports. In this paper we present an overview of an implementation of a Business Intelligence project in a national company, the problems we confronted with and the techniques that we applied to reduce the cost of execution for improving query performance in this decisional support system.

Key-Words: Tuning and optimization, SQL query plans, Business Intelligence projects, Virtual data warehouse, Data extraction, Query optimization and performance, Partitioning techniques, Indexes, Analytical functions

1 Introduction

The main goal of Business Intelligence Systems (BIS) is to assist managers, at different levels in the organization, in taking decisions and to provide in real time representative information, to help and support them in their activities such as analyzing departmental data, planning and forecasting activities for their decision area [1].

Through Business Intelligence Systems managers can manipulate large sets of data in a short period of time or in real time manner. In essence, managers at every departmental level can have a customized view that extracts information from transactional sources and summarizes it into meaningful indicators. These systems usually work with large sets of data and require a short response time and if you consider using analytical tools like OLAP against virtual data warehouses then you have to build your system through SQL queries and retrieve data directly from OLTP systems. In this case, the large amount of data in ERP systems may lead to an increase of responding time for BIS. That's why you should consider applying optimization techniques in order to improve the BI system's performance.

2 Data warehouses – a different business perspective

ERP systems are implemented in many organizations for operational and transactional processing for different functional areas such as: financials, inventory, purchase, order management, production. Information from these functional areas within an ERP system is managed by a relational software database (such as Oracle Database, DB2 or Microsoft SQL Server). Operational levels of management require detailed reports with daily operational activities. But the managers need information for strategic and tactical decision that often requires reports with aggregated data from ERP and non-ERP application sources. The usual reports developed from daily transactions does not satisfy the business needs, an executive cannot take a real time decision based on a hundred pages per month cash-flow detailed report. Information must be aggregated and presented with a template based on a business model. In the table below we represent the main differences between ERP reports and BIS reports:

CHARACTERISTICS	ERP REPORTS	EIS REPORTS
Objectives	Analyze indicators that measure current and internal activities or daily reports	Processes optimization, analyze key performance indicators, forecast internal and external data, internal and external focus
Level of decision	Operational/Medium	Strategic/High
User involved	Operational level of management	Executives, strategic level of management
Data Management	Relational databases Data warehouse	Data warehouse OLAP Data Mining
Typical operation	Report/Analyze	Analyze
Number of records/transaction	Limited	Huge
Data Orientation	Record	Cube
Level of detail	Detailed, summarized, pre-aggregate	Aggregate
Age of data	Current	Historical/current/prospective

Table 1: A comparison between ERP and BIS reports

The reports developed on transactional data usually operate on relatively few rows at a time. So, in order to improve the query performance we can use an index that can point to the rows that are required and, for this case the DBMS system can construct an accurate plan to access those rows efficiently through the shortest possible path. In Business Intelligence environments selectivity is less important, because they often access more table's rows and full table scans are common. In this situation indexes are not even used [4]. BI Systems usually work with large sets of data and require a short response time. If you consider not using analytical tools like OLAP and data warehousing techniques then you have to build your system through SQL queries and retrieve data directly from OLTP systems. In this case, the large amount of data in ERP systems may lead to an increase of responding time for the BIS. That's why you should consider phrasing the queries using the best optimization techniques.

2.1 Data warehouse - virtual vs. aggregate

From our point of view a data warehouse represents the main technique for integrating and organizing data from different sources, homogeneous or heterogeneous, with different characteristics, extracted from transactional systems or from flat files. Data warehouses integrate data based on a business model, and contain an extract, transform and load process (ETL) that is applied in order to aggregate and store data on several hierarchical levels required for Business Intelligence systems' dynamical extraction and analysis.

From this approach, the role of the data warehouse is essential for data management and knowledge management. But, in data warehouse's development in the organization or even at departmental levels (also known as data marts) the main problems are about online performance, data access, metadata administration, storing and organizing the data warehouse, the size and the periodical loading of data. These problems may appear distinctively accordingly to the characteristics and the type of the data warehouse and also to the performance of the DBMS in charge.

In the following sections we'll present a classification of the data warehouse from the functional area point of view: there are three types of data warehouses [2], [3]:

- *Central data warehouse or enterprise warehouse* collects all the data from the organization that concern different business subjects for taking decisions and contains a large amount of data. Usually contains both detailed data and aggregate data and in terms of size it starts from a few gigabytes to several terabytes. This type of data warehouse must be implemented on powerful UNIX servers or against parallel architectures and requires lots of resources and expenses, the development phases take more than one year.

- *Data mart* contains a subset of organizational data, with specific characteristics required by a group of users or managers or department. The data mart's domain is limited on certain subjects and the data is usually aggregated on the hierarchical levels developed for these specific requirements. The data marts are implemented at departmental levels with fewer resources than enterprise data warehouses and the platforms are UNIX or Windows 2000/2003. The development cycle takes several months or about a year and the size counts over 1 gigabyte. The data mart is more easily to implement and can be considered as a sub-data warehouse.
- *Virtual data warehouse* is a set of views or materialized views among the operational databases logically implemented as a data warehouse (in terms of dimensions and facts). In order to improve the query performance only a few of the view are materialized and aggregated. A virtual data warehouse is easy to developed, but the problem of extracting data and processing is the task of the DBMS, this can lead to inefficiency and low-performance and also time consuming. But the necessity of storing data into a data warehouse is eliminated. The virtual data warehouse involves a smaller size of data, if this size is increasing the extracting time is also increasing and our recommendation is to combine the virtual data warehouse with a solution that can store the aggregate data (a data mart or a data warehouse).

In the next table we compare from different pot of view and criteria the advantages and disadvantages of these three types of data warehouses.

Criteria	**Enterprise data warehouse**		**Data Mart**	
	Virtual	**Aggregate**	**Virtual**	**Aggregate**
The size of the data warehouse	None, data are stored in databases	Extremely high (terra bytes), data is aggregated and stored at every level	None, data are stored in databases	High (max 1Tb), data is aggregated and stored at every level but the size is smaller
The size of the data sources	Extremely high, data dictionary contains information about the VDW views an structures also	The DW not implies the data sources, the size depends on the transactional systems	Extremely high, data dictionary contains information about the DM views an structures also	The DM not implies the data sources, the size depends on the transactional systems
Data warehouse's objects	These are objects of the database – views and tables that will be mapped on the VDW	These are multidimensional objects: facts, dimensions, hierarchical levels, attributes, relations, etc.	These are objects of the database – views and tables that will be mapped on the VDM	These are multidimensional objects: facts, dimensions, hierarchical levels, attributes, relations, etc.
Metadata administration	It's realized at relational level.	It's realized at multidimensional level.	It's realized at relational level.	It's realized at multidimensional level.
Loading performance	Very low, it is not a loading it actually a query against millions of rows from relational databases.	Moderate, the data volume and structure make it time consuming and it can take more than a day.	Low, it is not a loading but a query against thousands of rows from relational databases.	High, the volume of data it is not so big so the process can take less than a day.
ETL process	It is realized at the relational level and consists in data processing through	It is realized at the data warehouse level through functions, procedures and special	It is realized at the relational level and consists in data processing through	It is realized at the data warehouse level through functions,

	functions, packages and stored procedures. The process is realized in online with low performance.	operators. It a very complex process for data transformation.	functions, packages and stored procedures. The process is realized in online with low performance.	procedures and special operators. It a complex process for data transformation.
The level of detail	Detailed, the data are extracted directly from relational sources.	Detailed and aggregate, data is stored pre-calculated.	Detailed, the data are extracted directly from relational sources.	Detailed and aggregate, data is stored pre-calculated.
Operators and analytical processing	Usually are applied at the interface level or at the client level.	They are applied at the data warehouse level and at the applications level.	Usually are applied at the interface level or at the client level.	They are applied at the data warehouse level and at the applications level.
Modifying the DW's structure	Very difficult, with important changes in VDW's views and structures – is the rebuilding of the VDW.	With difficulties, it implies several changes in DW's structure. At the development phase it should be considered such possibilities in order not to re-build the whole DW.	Difficult, with several changes in the VDM's structure and views.	Easily, not so many changes.
Analytical performance	Very low, all the aggregations, sorting and calculations are processed online, that lead to an increasing time, over an hour.	High, all the aggregations, sorting and calculations are processed before the extraction and stored in that form and retrieved by the client application. But the size of the DW can slow down the analytical process.	Moderate, all the aggregations, sorting and calculations are processed online, but the size of the data is not so huge and lead to respond time in term of minutes.	Very high, all the aggregations, sorting and calculations are processed before the extraction and stored in that form and retrieved by the client application.
Historical data analysis	The analysis is limited to the period of relational sources.	Data are loaded into the DW periodically, so the historical data is available from the beginning of the DW.	The analysis is limited to the period of relational sources.	Data are loaded into the DM periodically, so the historical data is available from the beginning of the DM.
Forecast	It can be applied only at the client level, but it's difficult to work with due of the huge volume of data.	It can be applied both at the data warehouse level and at the client level using special functions and operators.	It can be applied only at the client level, and it's working well with short periods.	It can be applied both at the data warehouse level and at the client level using special functions and operators.
Application independence	The impact of data changes is very strong and the applications must be re-build.	The logical levels of the data warehouse confer independence to the applications and the changes in the DW have a minor impact on them.	Applications are dependent on database views so every change in the VDM structure has an impact on the applications.	The logical levels of the data mart confer independence to the applications and the changes in the DW have a minor impact on them.
Development cycle	The development time is medium; the problems appear at the ETL	The development period can take over an year, the DW's objects must be design, the	The development period can take less than 3 months.	The development time is medium; the problems appear at the ETL

	process.	ETL process must be developed, also mappings and transformations take place.		process.
Technical risks	The slow respond time and time consuming queries, ETL process that must be realized at the client level, huge data volume, all these can make the VDW inefficient.	The complexity of the development cycle and the huge volume of data, also the impact over the organization are the major risks factors.	The ETL process that is realized at the client level.	There are fewer risks involved, but the functionality is not so proper developed.

Table 2: Comparative analysis of different types of data warehouses

3 Actual state and problems of the BI project

In the research project that we implemented in one of the multinational companies from our country we applied these concepts and also the requests from the executives and managers from the company. For the project lifecycle we applied the steps described in the book Executive Information Systems [2]. With BI techniques, like data warehouse, OLAP, data mining, portal we succeeded to implement the BI system's prototype and to validate it with the managers and executives. The BIS gathers data from the ERP system implemented in the organization from different functional areas or modules such as: financials, inventory, purchase, order management, production. Information from these functional modules within the ERP system is managed by a relational database management system - Oracle Database Enterprise Edition 10g. From the relevant data sources we applied an ETL (extract, transform and load) process that load data into the target sources. In this step we have to choose between the two data warehouses solutions: store data and virtual extraction. After a comparative analysis between these techniques we choose the second solution. The major elements in this choose ware that the ERP system was not yet fully implemented and there are many more changes to do, the amount of data is not so large – 3 millions records from January 2007 to August 2007, and the implementation of a virtual data warehouse is fastest and with a low budget then a traditional data warehouse. We also consider developing in parallel in the next months a traditional data warehouse after testing and implemented the actual prototype.

So, we developed the virtual data warehouse based on a set of views that collects data from the ERP system based on an ETL process that we designed. After we developed the analytical decisional reports and test them in a real organizational environment with over 100 users, we measured the performance of the system and the main problem was the high cost of execution. These analytical reports were over 80% resource consuming of the total resources allocated for the ERP and BI systems. Also, the critical moment when the system was breaking down was at the end of each month when all transactions from functional modules were posted to the General Ledger module. Testing all parameters and factors we concluded that the major problem was in the data extraction from the views.

First solution was to rewrite the views and build materialized views and semi-aggregate tables. Data sources are loaded in these tables by the ETL process periodically, at the end of the month after posting to the General Ledger or at user request. The ETL process contains a set of procedures grouped in three types of packages: extraction, transforming and loading. The target data are stored in tables used directly by the analytical reports. A benefit of this solution is that it eliminates the multiple joins from the views and also that we can use the ETL process to load data in the future data warehouse that we are going to implement in the next months.

After these operations we re-test the systems under the real conditions. The time for data extraction was again too long and the costs of executions consumed over 50% of total resources. So we consider using some of optimization techniques like: table partitioning, indexing, using hints and using analytical functions instead of data aggregation in some reports. In the following section we describe these techniques and provide a comparative analysis of some of our testing results.

3. Compute statistics and explain the execution plan of SQL queries

When a SQL statement is executed on an Oracle database, the Oracle query optimizer determines the most efficient execution plan after considering many factors related to the objects referenced and the conditions specified in the query. The optimizer estimates the cost of each potential execution plan based on statistics in the data dictionary for the data distribution and storage characteristics of the tables, indexes, and partitions accessed by the statement and it evaluates the execution cost. This is an estimated value depending on resources used to execute the statement. The optimizer calculates the cost of access paths and joins orders based on the estimated computer resources, which includes I/O, CPU, and memory [4]. This evaluation is an important factor in the processing of any SQL statement and can greatly affect execution time.

During the evaluation process, the query optimizer reviews statistics gathered on the system to determine the best data access path and other considerations. We can override the execution plan of the query optimizer with hints inserted in SQL statement. A SQL statement can be executed in many different ways, such as full table scans, index scans, nested loops, hash joins, and sort merge joins. We can set the parameters for query optimizer mode depending on our goal. By default the optimizer is set to the best throughput which chooses the least amount of resources necessary to process all rows accessed by the statement. But for MIS, time is one of the most important factor and we should optimize a statement with the goal of best response time. To set up the goal of the query optimizer we can use one of the hints that can override the OPTIMIZER_MODE initialization parameter for a particular SQL statement [4]. So, we can use FIRST_ROWS (n) hint to instruct Oracle to optimize an individual SQL statement with a goal of best response time to return the first n number of rows. The hint uses a cost-based approach for the SQL statement, regardless of the presence of statistic. The second option is to use ALL_ROWS hint that explicitly chooses the cost-based approach to optimize a SQL statement with a goal of best throughput.

We can collect exact or estimated statistics about physical storage characteristics and data distribution in these schema objects by using the DBMS_STATS package. We can use this package to collect histograms for table columns that contain values with large variations in number of duplicates, called skewed data [5]. The resulting statistics provide information about data uniqueness and distribution and based on this, the query optimizer can compute plan costs with a high degree of accuracy. This enables the query optimizer to choose the best execution plan based on the least cost. For example we can gather and view statistics against tables in our schema:

```
        BEGIN
        DBMS_STATS.GATHER_TABLE_STATS
(USER, 'BALANCE_RESULTS_A');
        END;
        /
        SELECT COLUMN_NAME, COUNT(*)
        FROM USER_TAB_HISTOGRAMS
        WHERE TABLE_NAME='
BALANCE_RESULTS_A'
        GROUP BY COLUMN_NAME;
```

4 Optimization solutions

4.1 Partitioning

The main objective of Partitioning technique is to radically decrease the amount of disk activity and limiting the amount of data to be examined or operated on and enabling parallel execution required to perform Business Intelligence queries against virtual data warehouses. Tables are partitioning using a partitioning key that is a set of columns which will determine by their conditions in which partition a given row will be store. Oracle Database 10g on which our ERP is implemented provides three techniques for partitioning tables:

Range Partitioning - specify by a range of values of the partitioning key;

List Partitioning - specify by a list of values of the partitioning key;

Hash Partitioning - a hash algorithm is applied to the partitioning key to determine the partition for a given row;

Also, there can be use sub partitioning techniques in which the table in first partitioned by range/list/hash and then each partition is divided in sub partitions:

Composite Range-Hash Partitioning – a combination of Range and Hash partitioning techniques, in which a table is first range-partitioned, and then each individual range-partition is further sub-partitioned using the hash partitioning technique;

Composite Range-List Partitioning - a combination of Range and List partitioning techniques, in which a table is first range-partitioned, and then each individual range-partition is further sub-partitioned using the list partitioning technique.

Index-organized tables can be partitioned by range, list, or hash [4]

In our case we consider evaluating each type of partitioning technique and choose the best method that can improve the BI system's performance.

Thus, we create two tables based on the main table which is used by some analytical reports and compare the execution cost obtained by applying the same query on them. First table BALANCE_RESULTS_A contained un-partitioned data and is the target table for an ETL sub-process. It counts 55000 rows and the structure is shown below in the scripts. The second table BALANCE_RESULTS_B is a range partitioned table by column ACC_DATE which refers to the accounting date of the transaction. This table has four partitions as you can observe from the script below:

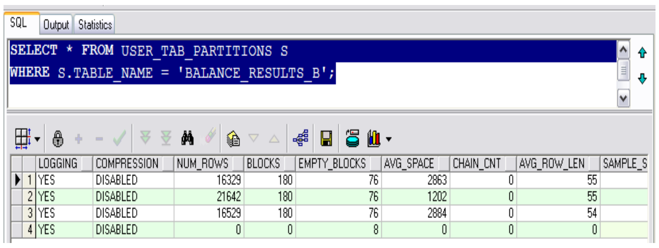
```
CREATE TABLE BALANCE_RESULTS_B
( ACC_DATE     DATE NOT NULL,
  PERIOD VARCHAR2(15) NOT NULL,
  ACC_D  NUMBER,
  ACC_C  NUMBER,
  ACCOUNT     VARCHAR2(25),
  DIVISION  VARCHAR2(50),
  SECTOR   VARCHAR2(100),
  MANAGEMENT_UNIT VARCHAR2(100))
PARTITION BY RANGE (ACC_DATE)
(PARTITION QT1  VALUES LESS THAN
(TO_DATE('01-APR-2007', 'DD-MON-YYYY')),
PARTITION QT2 VALUES LESS THAN
(TO_DATE('01-JUL-2007', 'DD-MON-YYYY')),
PARTITION QT3  VALUES LESS THAN
(TO_DATE('01-OCT-2007', 'DD-MON-YYYY')),
PARTITION QT4  VALUES LESS THAN
(TO_DATE('01-JAN-2008', 'DD-MON-YYYY')));
```

Then, we create the third table which is partitioned and that contained also for each range partition four list partitions on the column "Division" which is very much used in data aggregation in our analytical reports. The script is showed below:

```
CREATE TABLE BALANCE_RESULTS_C
(  ACC_DATE     DATE NOT NULL,
  PERIOD VARCHAR2(15) NOT NULL,
  ACC_D  NUMBER,
  ACC_C  NUMBER,
  ACCOUNT     VARCHAR2(25),
  DIVISION  VARCHAR2(50),
  SECTOR   VARCHAR2(100),
  MANAGEMENT_UNIT VARCHAR2(100))
PARTITION BY RANGE (ACC_DATE)
SUBPARTITION BY LIST (DIVISION)
(PARTITION QT1  VALUES LESS THAN
(TO_DATE('01-APR-2007', 'DD-MON-YYYY'))
(SUBPARTITION QT1_OP VALUES
('A.MTN','B.CTM','C.TRS','D.WOD','E.DMA'),
 SUBPARTITION QT1_GA VALUES ('F.GA
OP','G.GA CORP'),
 SUBPARTITION QT1_AFO VALUES ('H.AFO
DIV','I.AFO CORP'),
 SUBPARTITION QT1_EXT VALUES
('J.EXT','K.IMP') ),
PARTITION QT2 VALUES LESS THAN
(TO_DATE('01-JUL-2007', 'DD-MON-YYYY'))
(SUBPARTITION QT2_OP VALUES
('A.MTN','B.CTM','C.TRS','D.WOD','E.DMA'),
 SUBPARTITION QT2_GA VALUES ('F.GA
OP','G.GA CORP'),
 SUBPARTITION QT2_AFO VALUES ('H.AFO
DIV','I.AFO CORP'),
 SUBPARTITION QT2_EXT VALUES
('J.EXT','K.IMP')),
PARTITION QT3  VALUES LESS THAN
(TO_DATE('01-OCT-2007', 'DD-MON-YYYY'))
(SUBPARTITION QT3_OP VALUES
('A.MTN','B.CTM','C.TRS','D.WOD','E.DMA'),
 SUBPARTITION QT3_GA VALUES ('F.GA
OP','G.GA CORP'),
 SUBPARTITION QT3_AFO VALUES ('H.AFO
DIV','I.AFO CORP'),
 SUBPARTITION QT3_EXT VALUES
('J.EXT','K.IMP')),
PARTITION QT4  VALUES LESS THAN
(TO_DATE('01-JAN-2008', 'DD-MON-YYYY'))
(SUBPARTITION QT4_OP VALUES
('A.MTN','B.CTM','C.TRS','D.WOD','E.DMA'),
 SUBPARTITION QT4_GA VALUES ('F.GA
OP','G.GA CORP'),
 SUBPARTITION QT4_AFO VALUES ('H.AFO
DIV','I.AFO CORP'),
 SUBPARTITION QT4_EXT VALUES
('J.EXT','K.IMP')));
```

After loading data in these two partitioned tables we gather statistics with Analyze clause and the results of these statistics are showed below in fig 1 and fig 2:

```
SELECT * FROM USER_TAB_PARTITIONS S
WHERE S.TABLE_NAME =
'BALANCE_RESULTS_B';
```

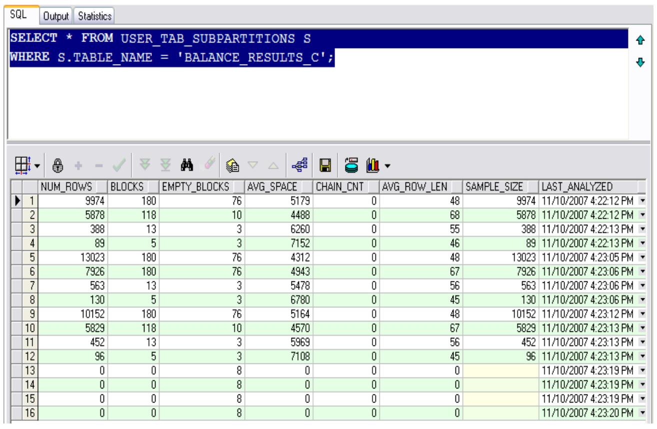


```
SELECT * FROM USER_TAB_PARTITIONS S
WHERE S.TABLE_NAME = 'BALANCE_RESULTS_B';
```

	LOGGING	COMPRESSION	NUM_ROWS	BLOCKS	EMPTY_BLOCKS	AVG_SPACE	CHAIN_CNT	AVG_ROW_LEN	SAMPLE_S
1	YES	DISABLED	16329	180	76	2863	0	55	
2	YES	DISABLED	21642	180	76	1202	0	55	
3	YES	DISABLED	16529	180	76	2884	0	54	
4	YES	DISABLED	0	0	8	0	0	0	

Fig. 1: Statistics from table BALANCE_RESULTS_B

```
SELECT * FROM USER_TAB_SUBPARTITIONS S
WHERE S.TABLE_NAME =
'BALANCE_RESULTS_C';
```



```
SELECT * FROM USER_TAB_SUBPARTITIONS S
WHERE S.TABLE_NAME = 'BALANCE_RESULTS_C';
```

	NUM_ROWS	BLOCKS	EMPTY_BLOCKS	AVG_SPACE	CHAIN_CNT	AVG_ROW_LEN	SAMPLE_SIZE	LAST_ANALYZED
1	9974	180	76	5179	0	48	9974	11/10/2007 4:22:12 PM
2	5878	118	10	4488	0	68	5878	11/10/2007 4:22:12 PM
3	388	13	3	6260	0	55	388	11/10/2007 4:22:13 PM
4	89	5	3	7152	0	46	89	11/10/2007 4:22:13 PM
5	13023	180	76	4312	0	48	13023	11/10/2007 4:23:05 PM
6	7926	180	76	4943	0	67	7926	11/10/2007 4:23:06 PM
7	563	13	3	5478	0	56	563	11/10/2007 4:23:06 PM
8	130	5	3	6780	0	45	130	11/10/2007 4:23:06 PM
9	10152	180	76	5164	0	48	10152	11/10/2007 4:23:12 PM
10	5829	118	10	4570	0	67	5829	11/10/2007 4:23:13 PM
11	452	13	3	5969	0	56	452	11/10/2007 4:23:13 PM
12	96	5	3	7108	0	45	96	11/10/2007 4:23:13 PM
13	0	0	8	0	0	0		11/10/2007 4:23:19 PM
14	0	0	8	0	0	0		11/10/2007 4:23:19 PM
15	0	0	8	0	0	0		11/10/2007 4:23:19 PM
16	0	0	8	0	0	0		11/10/2007 4:23:20 PM

Fig. 2: Statistics from table BALANCE_RESULTS_C

Analyzing the decision support reports we choose a sub-set of queries that are always performed and which are relevant for testing the optimization techniques. We run these queries on each test table A, B and C and compare the results in table 3.

TABLE:	**TABLE_A**	**TABLE_B**		**TABLE_C**		
	Not partitioned	Partition range by date on column “ACC_DATE”		Partition range by date with four list partitions on column “DIVISION”		
QUERRY:		Without partition clause	Partition (QT1)	Without partition clause	Partition (QT1)	Sub-partition (QT1_AFO)
SELECT * FROM …	100	121	-	224	-	-
WHERE EXTRACT (**MONTH FROM** ACC_DATE) **=1**;	103	124	42	227	72	72
… **AND** DIVISION='H.AFO DIVIZII'	101	122	42	10	5	72
SELECT SUM(ACC_D) TD, **SUM**(ACC_C) TC **FROM** BALANCE_RESULTS_A A **WHERE** EXTRACT (**MONTH FROM** ACC_DATE) =1 **AND** DIVISION='H.AFO DIVIZII'	101	122	42	10	5	72
… AND MANAGEMENT_UNIT ='MTN'	101	122	42	225	72	72
SELECT /*+ *USE_HASH (A U)**/ A.*, U.**LOCATION**,U.COUNTRY, U.REGION **FROM** BALANCE_RESULTS_A A ,MANAGEMENT_UNITS U **WHERE** A.MANAGEMENT_UNIT=U.MANAGEMENT_UNIT **AND** EXTRACT (**MONTH FROM** ACC_DATE) =1	106	127	46	231	76	76
… AND A.DIVISION = 'H.AFO DIVIZII'	105	126	45	14	9	75
…/*+ *USE_NL (A U)**/	191	212	71	100	8	101
…/*+ *USE_NL (A U)**/ --*WITH INDEXES*	151	172	58	60	6	88
…/*+ *USE_MERGE (A U)**/ --*WITH INDEXES*	104	125	45	13	8	75
…AND U.MANAGEMENT_UNIT ='MTN'	104	125	45	75	9	75

Thus, we create two tables based on the main table which is used by some analytical reports and compare the execution cost obtained by applying the same query on them. First table BALANCE_RESULTS_A contained un-partitioned data and is the target table for an ETL sub-process. It counts 55000 rows and the structure is shown below in the scripts. The second table BALANCE_RESULTS_B is a range partitioned table by column ACC_DATE which refers to the accounting date of the transaction. This table has four partitions as you can observe from the script below:

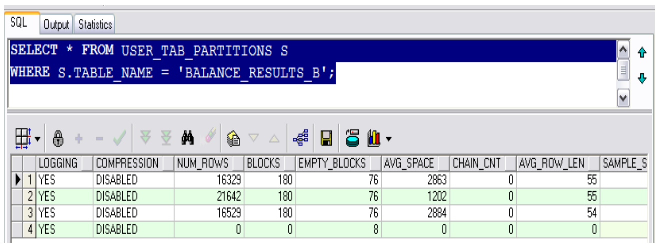
```
CREATE TABLE BALANCE_RESULTS_B
( ACC_DATE     DATE NOT NULL,
  PERIOD VARCHAR2(15) NOT NULL,
  ACC_D  NUMBER,
  ACC_C  NUMBER,
  ACCOUNT     VARCHAR2(25),
  DIVISION  VARCHAR2(50),
  SECTOR   VARCHAR2(100),
  MANAGEMENT_UNIT VARCHAR2(100))
PARTITION BY RANGE (ACC_DATE)
(PARTITION QT1  VALUES LESS THAN
(TO_DATE('01-APR-2007', 'DD-MON-YYYY')),
PARTITION QT2 VALUES LESS THAN
(TO_DATE('01-JUL-2007', 'DD-MON-YYYY')),
PARTITION QT3  VALUES LESS THAN
(TO_DATE('01-OCT-2007', 'DD-MON-YYYY')),
PARTITION QT4  VALUES LESS THAN
(TO_DATE('01-JAN-2008', 'DD-MON-YYYY')));
```

Then, we create the third table which is partitioned and that contained also for each range partition four list partitions on the column "Division" which is very much used in data aggregation in our analytical reports. The script is showed below:

```
CREATE TABLE BALANCE_RESULTS_C
(  ACC_DATE     DATE NOT NULL,
  PERIOD VARCHAR2(15) NOT NULL,
  ACC_D  NUMBER,
  ACC_C  NUMBER,
  ACCOUNT     VARCHAR2(25),
  DIVISION  VARCHAR2(50),
  SECTOR   VARCHAR2(100),
  MANAGEMENT_UNIT VARCHAR2(100))
PARTITION BY RANGE (ACC_DATE)
SUBPARTITION BY LIST (DIVISION)
(PARTITION QT1  VALUES LESS THAN
(TO_DATE('01-APR-2007', 'DD-MON-YYYY'))
(SUBPARTITION QT1_OP VALUES
('A.MTN','B.CTM','C.TRS','D.WOD','E.DMA'),
 SUBPARTITION QT1_GA VALUES ('F.GA
OP','G.GA CORP'),
 SUBPARTITION QT1_AFO VALUES ('H.AFO
DIV','I.AFO CORP'),
 SUBPARTITION QT1_EXT VALUES
('J.EXT','K.IMP') ),
PARTITION QT2 VALUES LESS THAN
(TO_DATE('01-JUL-2007', 'DD-MON-YYYY'))
(SUBPARTITION QT2_OP VALUES
('A.MTN','B.CTM','C.TRS','D.WOD','E.DMA'),
 SUBPARTITION QT2_GA VALUES ('F.GA
OP','G.GA CORP'),
 SUBPARTITION QT2_AFO VALUES ('H.AFO
DIV','I.AFO CORP'),
 SUBPARTITION QT2_EXT VALUES
('J.EXT','K.IMP')),
PARTITION QT3  VALUES LESS THAN
(TO_DATE('01-OCT-2007', 'DD-MON-YYYY'))
(SUBPARTITION QT3_OP VALUES
('A.MTN','B.CTM','C.TRS','D.WOD','E.DMA'),
 SUBPARTITION QT3_GA VALUES ('F.GA
OP','G.GA CORP'),
 SUBPARTITION QT3_AFO VALUES ('H.AFO
DIV','I.AFO CORP'),
 SUBPARTITION QT3_EXT VALUES
('J.EXT','K.IMP')),
PARTITION QT4  VALUES LESS THAN
(TO_DATE('01-JAN-2008', 'DD-MON-YYYY'))
(SUBPARTITION QT4_OP VALUES
('A.MTN','B.CTM','C.TRS','D.WOD','E.DMA'),
 SUBPARTITION QT4_GA VALUES ('F.GA
OP','G.GA CORP'),
 SUBPARTITION QT4_AFO VALUES ('H.AFO
DIV','I.AFO CORP'),
 SUBPARTITION QT4_EXT VALUES
('J.EXT','K.IMP')));
```

After loading data in these two partitioned tables we gather statistics with Analyze clause and the results of these statistics are showed below in fig 1 and fig 2:

```
SELECT * FROM USER_TAB_PARTITIONS S
WHERE S.TABLE_NAME =
'BALANCE_RESULTS_B';
```



```
SELECT * FROM USER_TAB_PARTITIONS S
WHERE S.TABLE_NAME = 'BALANCE_RESULTS_B';
```

	LOGGING	COMPRESSION	NUM_ROWS	BLOCKS	EMPTY_BLOCKS	AVG_SPACE	CHAIN_CNT	AVG_ROW_LEN	SAMPLE_S
1	YES	DISABLED	16329	180	76	2863	0	55	
2	YES	DISABLED	21642	180	76	1202	0	55	
3	YES	DISABLED	16529	180	76	2884	0	54	
4	YES	DISABLED	0	0	8	0	0	0	

Fig. 1: Statistics from table BALANCE_RESULTS_B

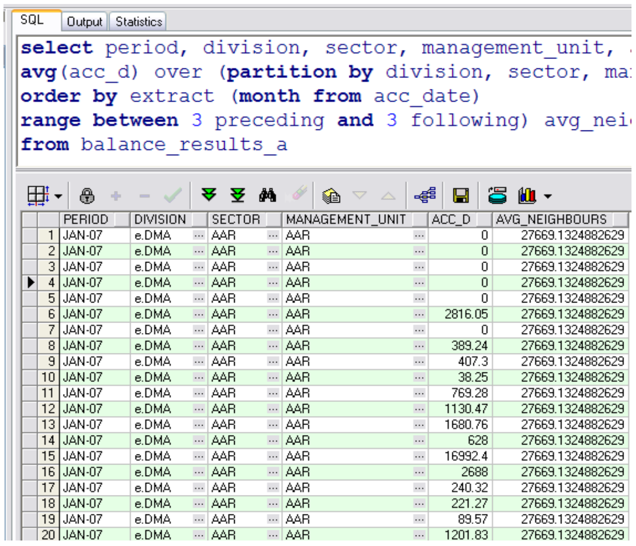
```
SELECT * FROM USER_TAB_SUBPARTITIONS S
WHERE S.TABLE_NAME =
'BALANCE_RESULTS_C';
```

Window sizes can be based on either a physical number of rows or a logical interval, based on conditions over values [1]

Analytic functions are performed after completing operations such joins, WHERE, GROUP BY and HAVING clauses, but before ORDER BY clause. Therefore, analytic functions can appear only in the select list or ORDER BY clause [4].

Analytic functions are commonly used to compute cumulative, moving and reporting aggregates. The need for these analytical functions is to provide the power of comparative analyses in the BI reports and to avoid using too much aggregate data from the virtual data warehouse. Thus, we can apply these functions to write simple queries without grouping data like the following example in which we can compare the amount of current account with the average for three consecutive months in the same division, sector and management unit, back and forward (fig 3):

SELECT PERIOD, DIVISION, SECTOR, MANAGEMENT_UNIT, ACC_D,
AVG(ACC_D) OVER (PARTITION BY DIVISION, SECTOR, MANAGEMENT_UNIT
ORDER BY EXTRACT (MONTH FROM ACC_DATE) RANGE BETWEEN 3 PRECEDING AND 3 FOLLOWING) AVG_NEIGHBOURS
FROM BALANCE_RESULTS_A



```
select period, division, sector, management_unit,
avg(acc_d) over (partition by division, sector, ma
order by extract (month from acc_date)
range between 3 preceding and 3 following) avg_nei
from balance_results_a
```

	PERIOD	DIVISION	SECTOR	MANAGEMENT_UNIT	ACC_D	AVG_NEIGHBOURS
1	JAN-07	e.DMA	AAR	AAR	0	27669.1324882629
2	JAN-07	e.DMA	AAR	AAR	0	27669.1324882629
3	JAN-07	e.DMA	AAR	AAR	0	27669.1324882629
4	JAN-07	e.DMA	AAR	AAR	0	27669.1324882629
5	JAN-07	e.DMA	AAR	AAR	0	27669.1324882629
6	JAN-07	e.DMA	AAR	AAR	2816.05	27669.1324882629
7	JAN-07	e.DMA	AAR	AAR	0	27669.1324882629
8	JAN-07	e.DMA	AAR	AAR	389.24	27669.1324882629
9	JAN-07	e.DMA	AAR	AAR	407.3	27669.1324882629
10	JAN-07	e.DMA	AAR	AAR	38.25	27669.1324882629
11	JAN-07	e.DMA	AAR	AAR	769.28	27669.1324882629
12	JAN-07	e.DMA	AAR	AAR	1130.47	27669.1324882629
13	JAN-07	e.DMA	AAR	AAR	1680.76	27669.1324882629
14	JAN-07	e.DMA	AAR	AAR	628	27669.1324882629
15	JAN-07	e.DMA	AAR	AAR	16992.4	27669.1324882629
16	JAN-07	e.DMA	AAR	AAR	2688	27669.1324882629
17	JAN-07	e.DMA	AAR	AAR	240.32	27669.1324882629
18	JAN-07	e.DMA	AAR	AAR	221.27	27669.1324882629
19	JAN-07	e.DMA	AAR	AAR	89.57	27669.1324882629
20	JAN-07	e.DMA	AAR	AAR	1201.83	27669.1324882629

Fig 3: Comparative costs obtained with analytical functions

5 Conclusions

Virtual data warehouse that we implemented in the Business Intelligence System developed in a multinational company is based on a set of views that extracts, joins and aggregates rows from many tables from the ERP system. In order to develop this decisional system we have to build analytical reports based on SQL queries. But the data extraction is the major time and cost consuming job. So, the solution is to apply a several techniques that improved the performance. In this paper we presented some of these techniques that we applied, like table partitioning, using hints, loading data from the online views in materialized views or in tables in order to reduce multiple joins and minimized the execution costs. Also, for developing reports easiest an important option is to choose analytic functions for predictions (LAG and LEAD), subtotals over current period (SUM and COUNT), classifications or ratings (MIN, MAX, RANK, FIRST_VALUE and LAST_VALUE). Another issue that is discussed in this article in the difference between data warehouses in which the data are stored in aggregate levels and virtual data warehouse. The comparative analysis on which we based in taking our choice between traditional data warehouse and virtual warehouse in the subject of another article that will be presented in the future conferences.

References:

[1] Lungu Ion, Bara Adela, Fodor Anca - *Business Intelligence tools for building the Executive Information Systems*, 5thRoEduNet International Conference, Lucian Blaga University, Sibiu, June 2006

[2] Lungu Ion, Bara Adela – *Executive Information Systems*, ASE Printing House, 2007

[3] Lungu Ion, Bara Adela, Diaconiţa Vlad - *Building Analytic Reports for Decision Support Systems - Aggregate versus Analytic Functions*, “Economy Informatics” Review, nr.1-4/2006, pg. 17-20, ISSN 1582-7941. www.ssrn.com

[4] Oracle Corporation - *Database Performance Tuning Guide 10g Release 2 (10.2)*, Part Number B14211-01, 2005

[5] Oracle Corporation - *Oracle Magazine*, 2006

[6] www.oracle.com

[7] Ralph Kimball - *The Data Warehouse Toolkit*, John Wiley & Sons, New York, 1996