'Shared-Nothing' Cloud Data Warehouse Architecture

Janina POPEANGĂ
University of Economic Studies, Bucharest, Romania
janina.popeanga@yahoo.com

Energy management systems from Romania do not have the capabilities of energy specific management due to lack of technology for real-time monitoring. As was the case in many other countries, the advent of smart metering technology will increase the level of energy data significantly. Therefore, the purpose of this paper is to present solutions that need to be taken to solve problems linked with the increasing amount of data recorded by sensors.
For a better demonstration of theoretical elements exposed, we considered a data warehouse specific to utility companies.
Section 2 of this article defines the three widely used parallel data warehouse architectures, while in Section 3 we clarify what architecture is suited to develop a data warehouse in the cloud. In the last part we transposed our tables in a "shared-nothing" architecture, trying to analyze queries performance.
Keywords: *Shared-nothing architecture, Data Warehouse, Replication, Distribution, Cloud*

1 Introduction

The amount of data being collected by utility companies has increased enormously in many countries with the advent of the smart metering technology. Before smart grids, utilities had collected from their customers data regarding their consumption on a monthly frequency. With the smart metering technology readings are taken at shorter intervals - every few seconds, so the increase is very significant.
The increasing volume of data and the variety of new data sources (devices and sensors) have created new challenges for utilities industry in terms of how to organize the amount of data from different sources in order to ease the data analysis.
Data warehouse is a centralized storage method of organizing data from different sources, undergo the process of extraction, transformation and loading, and of aggregate storing on hierarchical levels, data which are used in various complex processing and dynamic analysis. A data warehouse is hosted on an enterprise mainframe server or in the cloud.
Unlike operational systems, data structures in a data warehouse are optimized for dynamic queries and fast analysis. Data are historical and are updated at regular intervals, depending on the reporting requirements.
However, faced with the growing weight of explosive volumes of data and the expansive variety of data types, the capacity of a centralized data warehouse seems too limited.
Consider a data warehouse designated to customers application which permits them visualizing their consumption data over a period of time and several analysis on those data. (**Fig. 1**)
Consider that the data warehouse that we mentioned contains 3 dimensions and one fact table:
User – table which contains the main user information, like: UserID, User Name, Street, City, District, Region, Phone, Email, Income Level, House Area (square meters), Persons (number of persons in the house), El_Heaters (number of electric heaters in the home), HeatingH (heating time), Refrigerators (number of refrigerators), AirCond (number of reverse cycle air conditioners), HWLights (number of high-watt lights), HinHome (number of hours spent at home during the day), Requip (renewable energy equipment), Password.

Sensors – table which contains data about sensors: SensID (sensor ID), SensType (sensor type), Status (Down, Down (Partial), Down (Acknowledged), Warning, Up, Paused, Unusual, Unknown), Installation date, Last Revision.
Time – a dimension on data warehouse, since utilities will frequently want to aggregate about it. Full_Date, Time (hh:mm:ss), Hour, Day, Month, Quarter, Year, Day of Week, Holiday and Weekend flags are implemented in application by using the dimensional attributes.
In the **EnergyConsumption** fact table, energy consumption is measured below the level of calendar day, down to hour or minute or even seconds.

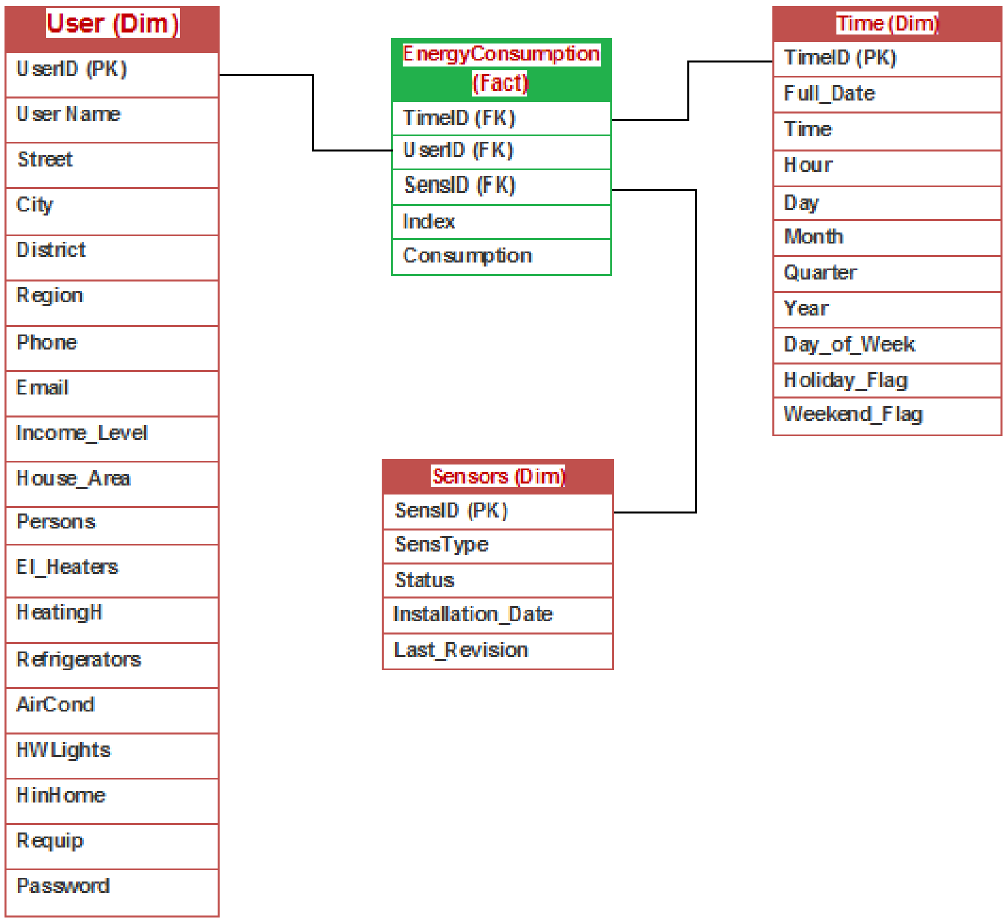


Fig. 1. Considered Energy Data Warehouse

The volume of data collected from consumers is increasing as time goes by, the number of concurrent queries is also rising.
The scalability issue becomes a huge challenge for centralized data warehouses. The solution addressing this dare is to distribute the large-scaled dataset and calculate the queries in parallel.
This paper focuses on the importance of solving these issues when creating a data warehouse in the cloud. The rest of the paper is structured as follows:

- Section 2 defines parallel data warehouse arhitechtures
- Section 3 discusses what should utilities look for in their cloud data warehouse
- Section 4 outlines how our tables should be split up across the nodes using a shared-nothing architecture – case study Microsoft SQL Server Parallel Data Warehouse (PDW)

Finally, we conclude this article in Section 5.

2 Better performance through Parallelism

Three widely used parallel hardware architectures for data warehousing exist, including shared-memory, shared-disk and shared-nothing. Consider also three basic elements in a parallel system: the central processing unit (CPU), the storage device (S) and memory (M).

The **Shared Memory (Shared Everything)** architecture is a system architecture where all existing CPUs share a global memory (M) and a single collection of disks (S). **(Fig. 2)**

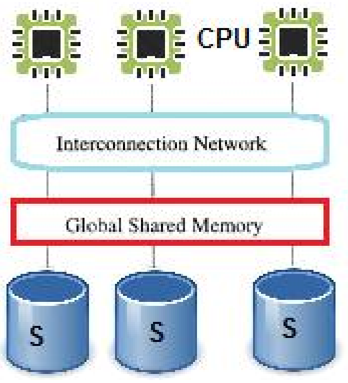


Fig. 2. Shared Memory

Only one Database Management System (DBMS) is present and can be executed in multiple processes or threads, in order to utilize all processors. [1]
Since there is a single memory, the lock manager and buffer poll are both stored there and this gives the chance to be easily accessed by all the CPUs.
There are two variations of Shared-Everything architecture [2]:

- The *symmetric multiprocessing architecture (SMP)*, where all the processors share a single pool of memory for read–write access concurrently and uniformly without latency.
- The *distributed shared memory architecture (DSM)*, where the latency to access memory depends on the relative distances of the processors and their dedicated memory pools.

The **Shared Disk** architecture is characterized by a number of independent processors (CPUs), each with its own memory (M), but a shared collection of disks (S) that is accessible to the DBMS of any Processing Node (PN). This means there is no longer a competition for shared memory, but for access to the shared disk. **(Fig. 3)**

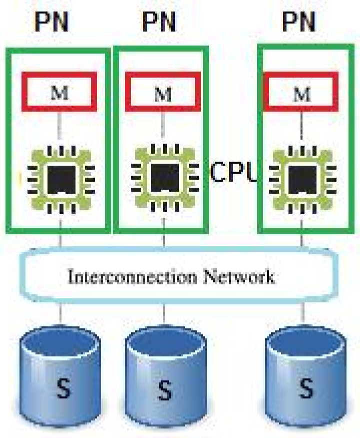


Fig. 3. Shared Disk

Since there is no pool of memory that is shared by all the CPUs, there is no place for the lock table or buffer pool to reside. To set locks, one must either centralize the lock manager on one processor or resort to a complex distributed locking protocol. [3]

The **Shared Nothing** architecture is a distributed computing architecture where nodes are networked to form a scalable system.
Each node has its own private memory (M), processor (CPUs) and storage devices (S) independent of any other node in the configuration. **(Fig. 4)** This means that every node stores its own lock table and buffer pool.
Such architectures are especially well suited to the star schema queries present in data warehouse workloads, as only a very limited amount of communication bandwidth is required to join one or more (typically

small) dimension tables with the (typically much larger) fact table. [3]
Data is horizontally partitioned across nodes, such that each node has a subset of the rows from each table that was distributed and all the replicated tables.

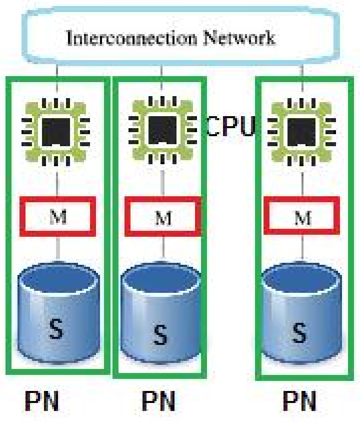


Fig. 4. Shared Nothing

The key feature of shared-nothing architecture is that the operating system not the application server owns responsibility for controlling and sharing hardware resources. [2]

3 What architecture is suited for a cloud data warehouse?

Cloud computing is probably the simplest and best fitted way for smart grids due to its scalable and flexible characteristics, and its capability to manage large amounts of data. [4]
Exactly these two major characteristics have contributed to the importance of parallel data warehousing:

- Data warehouses can become very large and exceedingly resource demanding
- queries and analyses must be answered within acceptable time limits.

One major advantage of shared-memory architecture is that the responsibility to handle the concurrency issues that result from the multiple parallel executions belongs to the operating system.

Unfortunately, shared-memory systems have fundamental scalability limitations, as all I/O and memory requests have to be transferred over the same bus that all of the processors share, causing the bandwidth of this bus to rapidly become a bottleneck. It is unusual to see shared-memory machines of larger than 8 or 16 processors unless they are custom-built from non-commodity parts, in which case they are very expensive. [3]
Therefore, there are significant scalability limits to any data warehouse based on shared-memory architecture.

One major advantage of shared-disk is that all the data is stored in the shared collection of disks. This means that there is no need to distribute parts of the data in each node.
It has the disadvantage of creating a possibly critical bottleneck and scalability limitations in the storage subsystem and interconnections, as all processing units share the same storage system. [1]
Therefore, shared disk architecture gives extremely limited capacity to scale.

Shared nothing does not typically have nearly as severe bus or resource contention as shared-memory or shared-disk machines, shared nothing can be made to scale to hundreds or even thousands of machines. Because of this, it is generally regarded as the best-scaling architecture [5].
Shared-nothing architecture scales better and is well suited for a cloud data warehouse considering very low-cost commodity PCs and networking hardware.
Nowadays, the popular distributed systems have almost all adopted the shared-nothing architectures, including peer-to- peer, cluster, Grid, and the Cloud. [6]
The *shared-nothing architecture* is used for overcoming the scalability limitations, to improve performance when loading and querying data concurrently as well as performing complex joins.

4 Parallel Data Warehouse

SQL Server Parallel Data Warehouse is a *Massively Parallel Processing* (MPP) solution, which means PDW uses a "*shared-nothing*" architecture, where there are multiple physical nodes, each running its own instance of SQL Server with dedicated CPU, memory, and storage.

PDW has two primary types of tables: *replicated* and *distributed*.

The purpose of **replicating tables** is to improve performance by keeping a copy of them on each compute node to support local joins, without having to handle complicated types of parallel queries or dimension-only queries between nodes. This type of table is most often used for dimension tables, but for a very large dimension table is recommended to use distribution. **(Fig. 5)**

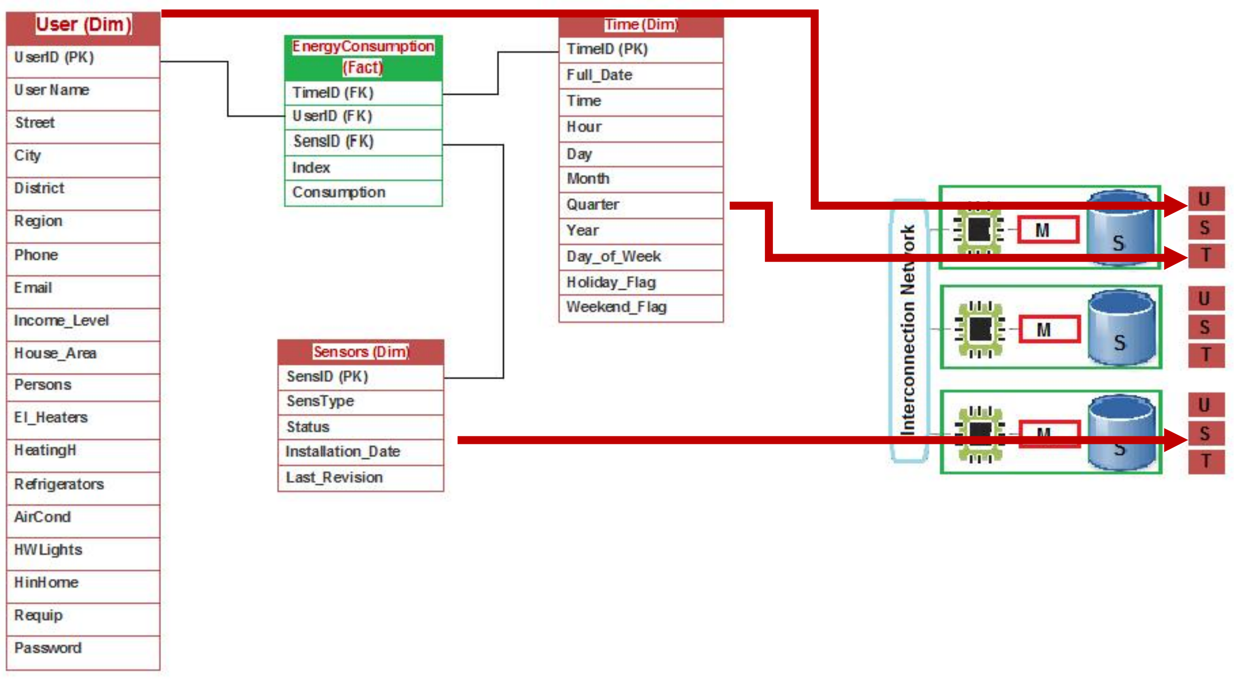


Fig.5. Dimension tables are replicated on every node

The code for replicating a dimension table from the Energy Data Warehouse looks like this:

```
CREATE TABLE User (
UserID varchar(10) NOT NULL,
UserName varchar(50),
Street varchar(50),
City varchar(20),
District varchar(15),
Region varchar(20),
Phone varchar(10),
Email varchar(50),
Income Level varchar(20),
House_Area int,
Persons int,
El_Heaters int,
HeatingH decimal(3,1),
Refrigerators int,
AirCond int,
HWLights int,
HinHome decimal(3,1),
Requip varchar(30),
Password varchar(10))
WITH
(DISTRIBUTION = REPLICATE);
```

If the distribution clause is omitted, the default is REPLICATE.

The same applies for creating the other dimension tables (Sensors and Time).

The purpose of **distribution** is to improve performance by spreading all the rows of a distributed table across all nodes, with the condition that each row from the source table ends up in only one node. **(Fig. 6)**

Large fact or transaction tables that contain billions or even trillions of rows are usually distributed.

The rows are mapped using a hash function on a distribution key from the table.

The distribution key must be a single attribute column, selected considering several criteria:

- High cardinality and even row counts – data must be distributed as evenly as possible across all the distributions on all nodes;

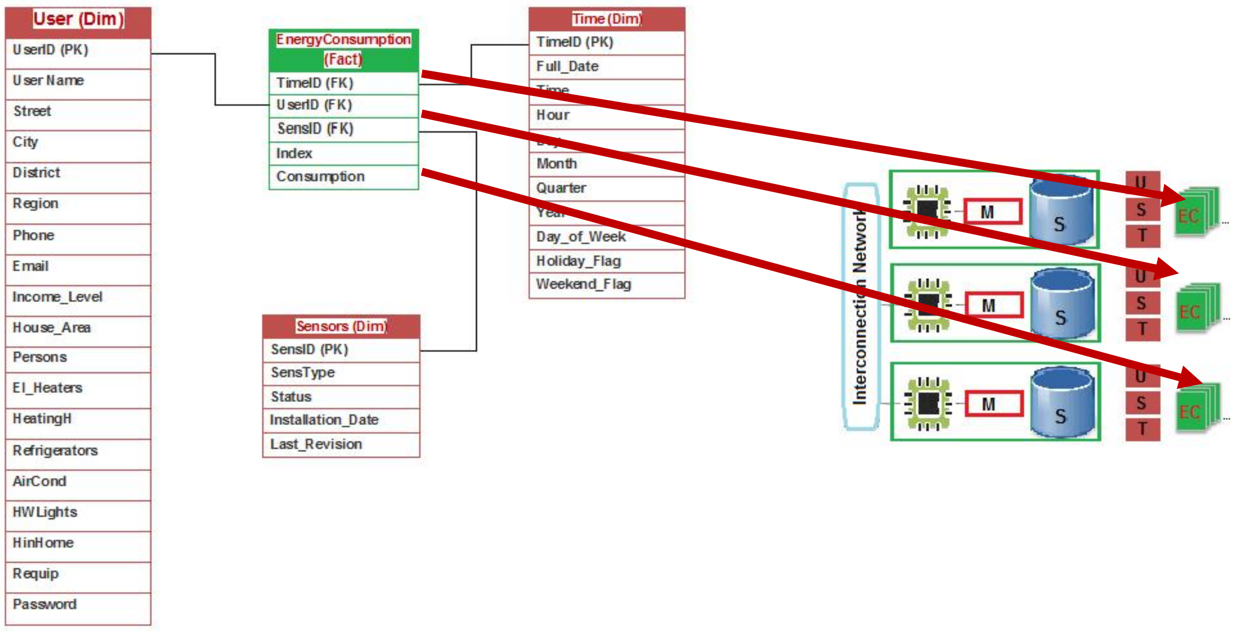


Fig. 6. Fact table is distributed on every node

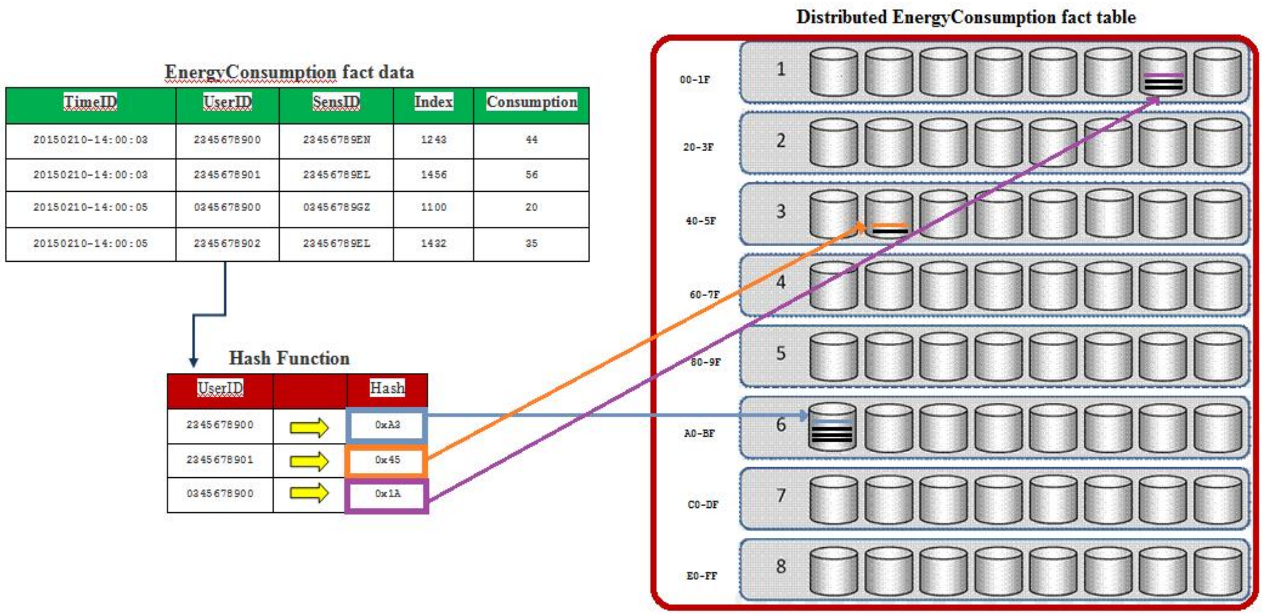
- Is recommended not to use as distributed key a column that is frequently restricted to a single value in queries.
- When distributing multiple fact tables, the needed analyzes are other considerations that must be taken into account.

  For example, by distributing multiple fact tables on the same distribution key, rows from the first table will be evenly distributed across all nodes and the rows from the other tables will be co-located on the same distribution. This way, queries that may need to join multiple distributed fact tables will perform fast.

The code for distributing the **EnergyConsumption** fact table from the Energy Data Warehouse, having **UserID** as the distribution key, looks like this:

```
CREATE TABLE EnergyConsumption (
TimeID varchar(20) NOT NULL,
UserID varchar(10) NOT NULL,
SensID varchar(10) NOT NULL,
Index decimal(10,3),
Consumption decimal(10,3))
WITH
(DISTRIBUTION = HASH(UserID));
```

The rows are mapped to the distributions using a hash function on the column that was chosen as the distribution key from the table, in our case **UserID**. **(Fig.7)**



EnergyConsumption fact data

TimeID	UserID	SensID	Index	Consumption
20150210-14:00:03	2345678900	23456789EN	1243	44
20150210-14:00:03	2345678901	23456789EL	1456	56
20150210-14:00:05	0345678900	03456789GZ	1100	20
20150210-14:00:05	2345678902	23456789EL	1432	35

Hash Function

UserID		Hash
2345678900	⇨	0xA3
2345678901	⇨	0x45
0345678900	⇨	0x1A

Fig. 7. Distributed **EnergyConsumption** fact table

The UserID key from each row from the EnergyConsumption fact data is passed through a Hash Function. The hashed values map to a single distribution on a single node.

For example, the row for UserID 0345678900 hashes to 0x1A, which maps to the second-last distribution of the first compute node. (Node:1, Distribution:7, Row:3)

The row for UserID 2345678901 hashes to 0x45, which maps to Node:3, Distribution: 2, Row:2.

After the tables are loaded with data, users will use SQL statements to query PDW tables.

For example, the following basic "shared-nothing" join runs against the distributed EnergyConsumption table created above and the replicated tables User and Time:

```
SELECT  u.UserID, sum(ec.Consumption)
FROM User u
JOIN EnergyConsumption ec ON u.UserID =
ec.UserID
JOIN Time t ON ec.TimeID = t.TimeID
WHERE u.Region = 'Oltenia' and t.Year =
'2014'
GROUP BY u.UserID
```

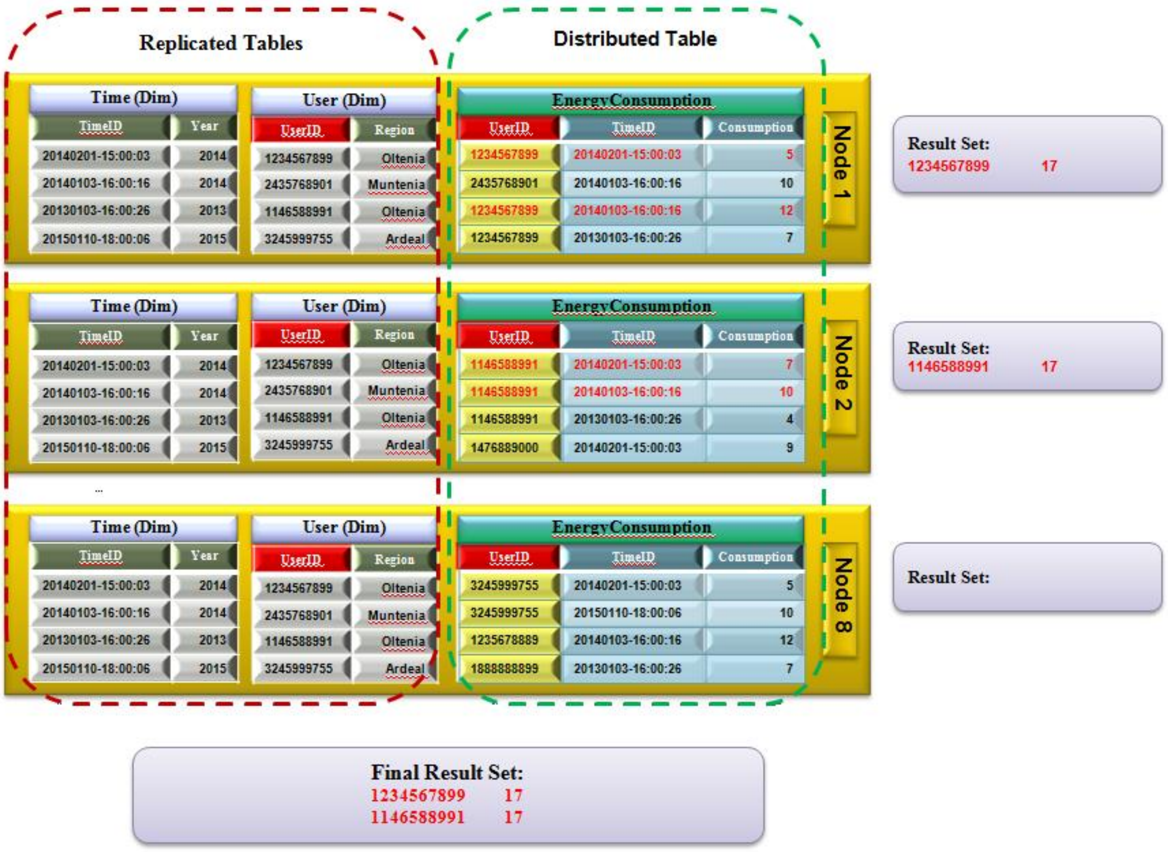
In this case user wants to find out the amount of energy consumption in 2004 for every consumer from Oltenia.

The query will be sent to each node, it will be executed in parallel on each compute node, then the PDW's MPP engine collects and merges all the parallel result sets from the nodes and sends back to the client a single result set. **(Fig. 8)**

This query performs very well even if EnergyConsumption is a very large distributed table. Giving the fact that the dimension tables are replicated on every node, each compute node can answer its part of the parallel query.

Some of the many benefits provided by the SQL Server Parallel Data Warehouse are *query performance* (10-100 times increased due to parallel execution), *data loading performance* (10-40 times faster due to parallel loading data), *the integration with cloud-born data* (Windows Azure HDInsight, Windows Azure blob storage) and *the HDInsight integration* into the PDW rack.

HDInsight integrates Hadoop within a parallel data warehouse processing (PDW). Scaling to the cloud is also easily enabled with the HDInsight Hadoop service on Windows Azure. [7]



Node 1

Time (Dim)

TimeID	Year
20140201-15:00:03	2014
20140103-16:00:16	2014
20130103-16:00:26	2013
20150110-18:00:06	2015

User (Dim)

UserID	Region
1234567899	Oltenia
2435768901	Muntenia
1146588991	Oltenia
3245999755	Ardeal

EnergyConsumption

UserID	TimeID	Consumption
1234567899	20140201-15:00:03	5
2435768901	20140103-16:00:16	10
1234567899	20140103-16:00:16	12
1234567899	20130103-16:00:26	7

Result Set:
1234567899 17

Node 2

Time (Dim)

TimeID	Year
20140201-15:00:03	2014
20140103-16:00:16	2014
20130103-16:00:26	2013
20150110-18:00:06	2015

User (Dim)

UserID	Region
1234567899	Oltenia
2435768901	Muntenia
1146588991	Oltenia
3245999755	Ardeal

EnergyConsumption

UserID	TimeID	Consumption
1146588991	20140201-15:00:03	7
1146588991	20140103-16:00:16	10
1146588991	20130103-16:00:26	4
1476889000	20140201-15:00:03	9

Result Set:
1146588991 17

Node 8

Time (Dim)

TimeID	Year
20140201-15:00:03	2014
20140103-16:00:16	2014
20130103-16:00:26	2013
20150110-18:00:06	2015

User (Dim)

UserID	Region
1234567899	Oltenia
2435768901	Muntenia
1146588991	Oltenia
3245999755	Ardeal

EnergyConsumption

UserID	TimeID	Consumption
3245999755	20140201-15:00:03	5
3245999755	20150110-18:00:06	10
1235678889	20140103-16:00:16	12
1888888899	20130103-16:00:26	7

Result Set:

Final Result Set:
1234567899 17
1146588991 17

Fig. 8. "Shared-Nothing" join

5 Conclusions

The volume of data collected by utility companies has increased a lot with the progress of the smart metering technology.

A scalable data warehouse architecture is critical, not only to manage very large volumes of data from different sources, but also to assure great response time to users.

Shared-memory and shared disk architectures cannot scale well with the rising amount of data, because these two architectures involve a large volume of data exchange over the interconnection network. But, the interconnection network cannot be infinitely extended, which becomes the main limitation of these two architectures. On the other hand, the shared-nothing architecture minimizes resource sharing.

Between the three basic parallel hardware architectures for data warehousing, "shared-nothing" architecture scales better and is well suited for a cloud data warehouse considering hardware and communication costs, better performance when loading and querying data simultaneously, as well when running complex queries.

Due to parallel execution of joins, query performance has increased up to 10-100 times compared to a join executed in a centralized data warehouse.

Also, due to parallel execution, data loading performance has increased up to 10-40 times compared to a centralized data warehouse.

In conclusion, benefits like storage of large amount of data from different sources (inclusive devices and sensors) and the fast query response times against those data are vital opportunities for competitive advantage for utilities companies.

Acknowledgment
This work was cofinanced from the European Social Fund through Sectoral Operational Programme Human Resources Development 2007-2013, project number POSDRU/159/1.5/S/142115 „Performance and excellence in doctoral and postdoctoral research in Romanian economics science domain”.

References

[1] P. Furtado, *A Survey on Parallel and Distributed Data Warehouses*, 2011, https://eden.dei.uc.pt/~pnf/publications/Furtado_survey.pdf
[2] K. Krishnan, Data Warehousing in the Age of Big Data, Publisher: Morgan Kaufmann, 2013, Print ISBN-13: 978-0-12-405891-0
[3] D. J. DeWitt, S.Madden and M. Stonebraker, *How To Build a High-Performance Data Warehouse*, 2009, http://db.csail.mit.edu/madden/high_perf.pdf
[4] J. POPEANGĂ, *Cloud Computing and Smart Grids*, Database Systems Journal vol. III, no. 3/2012.
[5] D. J. DeWitt and J. Gray, Parallel Database Systems: The Future of High Performance Database Processing, Communication of the ACM, Vol. 35(6), pp. 85-98, June 1992.
[6] Fr. Magoules, J. Pan and F. Teng, Cloud Computing: Data-Intensive Computing and Scheduling, Publisher: Chapman and Hall/CRC, 2012, Print ISBN-13: 978-1-4665-0782-1.
[7] Microsoft Corporation, *The Microsoft Modern Data Warehouse*, 2013, http://download.microsoft.com/download/C/2/D/C2D2D5FA-768A-49AD-8957-1A434C6C8126/The_Microsoft_Modern_Data_Warehouse_White_Paper.pdf

Janina POPEANGĂ graduated in 2010 the Faculty of Cybernetics, Statistics and Economic Informatics, Economic Informatics specialization. The title of her Bachelor’s thesis is “*Distributed Databases*”. In 2012, she graduated the Databases for Business Support master program with the thesis “*Monitoring and management of electric power consumption using sensorial data*”. Janina’s interests are broadly in the fields of databases and distributed systems. Since 2012 she is a Ph.D. Student in the Doctoral School of Bucharest Academy of Economic Studies. Her research focuses on real-time database systems, business intelligence analytics, sensor data management, smart grid and renewable energy.