Computing and Informatics, Vol. 42, 2023, 762–780, doi: 10.31577/cai_2023_3_762

OPTIMIZATION OF COLUMNAR NOSQL DATA WAREHOUSE MODEL WITH CLARANS CLUSTERING ALGORITHM

Nassima Soussi

LIPIM, National School of Applied Sciences
Sultan Moulay Slimane University
Khouribga, Morocco
e-mail: nassima.soussi@gmail.com

Abstract. In order to perfectly meet the needs of business leaders, decision-makers have resorted to the integration of external sources (such as Linked Open Data) in the decision-making system in order to enrich their existing data warehouses with new concepts contributing to bring added value to their organizations, enhance its productivity and retain its customers. However, the traditional data warehouse environment is not suitable to support external Big Data. To deal with this new challenge, several researches are oriented towards the direct conversion of classical relational data warehouse to a columnar NoSQL data warehouse, whereas the existing advanced works based on clustering algorithms are very limited and have several shortcomings. In this context, our paper proposes a new solution that conceives an optimized columnar data warehouse based on CLARANS clustering algorithm that has proven its effectiveness in generating optimal column families. Experimental results improve the validity of our system by performing a detailed comparative study between the existing advanced approaches and our proposed optimized method.

Keywords: Big Data, columnar NoSQL data warehouse, linked open data, clustering algorithms, Clarans

1 INTRODUCTION

Since many decades, data warehouse system has been a very important place in data analytics solutions thanks to its ability to effectively manage one of the major capital of any organization: the Data. However, the arrival of big data and the need for analysis continuously this voluminous mass in perpetual increase has impacted its ecosystem. This situation has led some analysts to anticipate the disappearance of DW in favor of Big Data systems, whereas the majority of data warehousing communities are campaigning for its extension [1] in order to align their decisional systems to Big Data requirements [2]. This alignment is materialized principally through the integration of Big Data pillars summarized in 5 Vs (Volume, Variety, Velocity, Value and Veracity) in the different phases of decision-making process.

In traditional data warehouses, a large number of requirements are not fully met by internal sources. This situation penalizes companies looking for real added value and contradicts the initial objective of decision-making systems. Hence, in order to help managers to make the right decisions, companies owning data warehousing technology have to enrich their existing data warehouses with new concepts by integrating external Big Data (related to their activities) in their decision-making system. However, traditional DW based on relational database is not suitable to support external big data characterized by its high volume and data format heterogeneity which requires a schemaless distributed system able to cohabit internal and external data efficiently contributing to the renaissance of this vital ecosystem.

To deal with the previous challenges, several researches that have been proposed in the literature are oriented towards the direct conversion of classical relational data warehouse to a columnar NoSQL data warehouse (CN-DW), whereas the few advanced works based on clustering algorithms are very limited and have several shortcomings. Thus, the issue of designing an optimized CN-DW still arises. In this paper, we revisit existing CN-DW models by treating its main limitations in our new solution that conceives an optimized model based on clustering algorithm [3] with an optimal column family's number. In addition, our system takes into consideration all relational data warehouse (Rel-DW) attributes to design a complete targeted model able to meet perfectly the new needs of business leaders.

The remainder of this paper is organized as follows: Section 2 presents the most recent related works in the current topic and discusses their main limitations. Section 3 highlights the main concepts used in the proposed solution, such as the HBase NoSQL database and Clarans clustering algorithms. Section 4 presents the functional architecture of our optimized CN-DW with a set of detailed algorithms for each phase. Section 5 improves the performance of our system with a real and practical comparison with the most recent existing approaches. Section 6 presents an analysis and discussion. Finally, Section 7 concludes our work and suggests some future extensions of this topic.

2 RELATED WORKS

In order to perfectly meet the needs of business leaders in the big data era, several researches have been made to ensure the enrichment of the decisional systems with different types of big data such as semantic web [4], social data [5, 6], NoSQL data [7], data lakes [8] and LOD [9, 10, 11]. However, neither of these approaches has implemented a decision-making system based on a big data warehouse, which is ready at all times to support the increased integration of big data so as to constantly satisfy the functional exigencies of decision-makers. To deal with this challenge, and to ensure a suitable environment for external big data, numerous works have been design a big data warehouse based on column oriented NoSQL database (CN-DW) from a classical DW (Rel-DW) trying to respect the maximum of specifications and peculiarities of each system. The existing works can be classified into three main categories:

Naive approaches: These approaches such as [12, 13, 14] and [15] are called naives since they propose a direct conversion method based on a set of mapping rules defined by matching the basic characteristics of each model. However, all these works neglected the consideration of any optimization operation; hence they have an imbalance on the number of attributes in column families. In addition, they did not control the number of generated column families.

Advanced approaches: In order to solve the shortcoming and weaknesses of naive approaches, several works have been made to enhance the existing CN-DW schema conception and group attributes in column families more efficiently. In [16], the authors decided to classify Rel-DW attributes into two column families in CN-DW schema: the first one gathers the most interrogated attributes, whereas the second one groups the remaining attributes. In addition, paper [17] presents a new method called NoSE (NoSQL Schema Evaluator) considered as a cost-based method aiming to recommend an optimized storage schemas for NoSQL column oriented database. However, the effectiveness of this system is limited to queries manipulating a small number of attributes. But all these approaches do not consider clustering techniques in their CN-DW conception methods to optimize the grouping of column families in order to improve the query processing performance and response times to complex queries.

Optimized approaches: In order to design the most optimized data model of CN-DW column families, the first work established in this direction is presented in [18] that propose a new method for grouping data carefully (from fact and dimensions of Rel-DW) in one CN-DW table using k-means as a clustering algorithm. In fact, the authors use this technique in order to group in the same column family the frequently used attributes based on a set of decisional queries. Likewise, the paper [19] addresses the same issue, with a normalized technique which consists to create deferent tables in CN-DW instead of one table. To do that, they grouped analogous queries in classes using k-medoid algorithm, end then they used the PSO algorithm (Particle Swarm Optimization algorithm based on meta-heuristics) to gather column families attributes according to each similar class of queries. The experimental results obtained by adopting clustering algorithms in CN-DW design prove the enhancement of decisional queries performance. Although, these optimized researches share some similarities with our solution presented in this paper, however they neglected two important points:

- The both clustering algorithms adopted above (k-means and k-medoid [20]) require the specification of clusters number without proposing any improvement of this initial configuration.
- They consider just the attributes used in decisional queries to conceive the distributed data warehouse. In this case, the designed CN-DW will be unable to meet new needs requiring the use of excluded attributes.

This comparative study showed that, despite these numerous solutions proposed in the literature (discussed previously), the issue of designing an optimized CN-DW still arises. To deal with the previous shortcomings, our current work provides a new approach aiming to design a CN-DW with an optimal clusters number with CLARANS algorithm. In addition, our system take into consideration all Rel-DW attributes to design a complete targeted model able to meet perfectly the new needs of business leaders.

Table 1 presents a technical comparison of the most recent approaches aiming to design CN-DW from Rel-DW. To realize this comparison, we considered the following main characteristics:

Conversion type: indicates the type of CN-DW design: (a) direct methods are based on a set of mapping rules defined by matching the basic characteristics of each model. (b) advanced methods are more developed than the first ones; for exemple they gather the most interrogated attributes in one columns, and the remaining attributes in the second one. (c) optimized methods design the most optimized data model of CN-DW.

Type of optimization technique used by optimized approaches.

Name of optimization technique

Complete model: in order to have a complete CN-DW that can meet future needs, it should contains all the Rel-DW attributes even if they are not used by current analytical questions.

Predefined CF number: it refers to the number of CN-DW column families: is it predefined randomly or generated and optimized by clustering algorithms.

Type of CN-DW: the NoSQL system used to implement the targeted CN-DW.

3 BACKGROUND

In this section, we introduce the main concepts used by our solution presented in Figure 1, that describes the proposed optimal CN-DW implemented in HBase as a column oriented NoSQL database and based on Clarans clustering algorithm to design the best regroupement of columns families.

	Conversion Type	**Type of Optimization Techniques**	**Name of Optimization Technique**	**Complete Model**	**Predefined CF Number**	**Type of CN-DW**
[**12**]	Direct			Yes	Yes	HBase
[**13**]	Direct			Yes	Yes	Hbase
[**14**]	Direct			Yes	Yes	HBase
[**15**]	Direct			Yes	Yes	Cassandra
[**16**]	Advanced			Yes	Yes	HBase
[**17**]	Advanced			Yes	Yes	Cassandra
[**19**]	Optimized	Clustering & meta-heuristic algorithms	K-medoid+ PSO	No	Yes	HBase
[**18**]	Optimized	Clustering algorithms	K-means	No	Yes	HBase

Table 1. Technical comparison of CN-DW design approaches

3.1 HBase

HBase [21] is a Column Oriented Database that looks remarkably like a relational database, but the concept is entirely different. It is dedicated to accommodating many columns (up to several million) for each line. It is characterized by a variable number of columns that can change from one row to another (we can consider that a column exists if it contains a value). This type of NoSQL database offers a high scalability in data storage and flexible schema due to the number of columns that can change from one row to another. The model of column-oriented database is composed of a set of Tables; each table contains a set of rows. Each row can be represented as $Ri = (Idi, (CFi1, CFi2, \ldots, CFim))$ with $i \in [1, n]$, $j \in [1, m]$, Idi is a row *id* and $CFij$ is a column family of the row Ri. Each column family can contain numerous columns that have the same categories of attributes which also called Column Qualifier. In this work we have chosen HBase to implement the CN-DW in order to have a flexible schema ready to receive heterogeneous external data easily and to improve the execution time of decision queries. In addition, we can operate on HBase tables with Spark and MapReduce for parallel processing of big data.

3.2 Clustering Algorithms

Automatic classification or clustering is an important step in the process of Knowledge Extraction from Data. It aims to develop an optimal partitioning by grouping data into classes that share similar characteristics where the data is generally represented by measurement vectors or points in a multidimensional space. Intuitively, vectors belonging to a valid cluster are more similar to each other than a vector belonging to a differ group. In other words, the goal of these methods is to identify groups from an unlabeled set of data vectors that share semantic similarities. This allows the user to construct a cognitive model, thus aiding the detection of the inherent structure of a data set.

There are four main families of clustering algorithms: Hierarchical algorithms, Algorithms by partition, Algorithms based on density, Classification based on quantification by grid. In this paper, the most suitable type for our problem is the data partitioning algorithms in order to design the best grouping of column families for the CN-DW. They consist in dividing directly a set of data (the attributes of Rel-DW) into k classes (or families of columns in our case) such that each class (or FC) must contain at least one attribute and each attribute must belong to a unique class unlike the so-called fuzzy classification which does not impose this condition. Among the famous algorithms of this type [22], we quote: K-means, K-medoids, Partition Around Medoid (PAM), Clustering LARge Applications (CLARA) et Clustering large applications based upon randomized search (CLARANS).

The general algorithm of a partition classification follows the steps below:

1. Determine the number of clusters.
2. Initialize cluster centers.
3. Partition the dataset.
4. Calculate cluster centers (make an update).
5. If the partitioning is unchanged (or the algorithm has converged), stop; otherwise go to step 3.

A problem that accompanies the use of a partition classification algorithm is the choice of the desired output classes number. Partition clustering techniques generally produce clusters by optimizing an objective function defined locally (on a subset of data vectors) or globally (defined on all vectors) which translates that the objects must be similar within a same class, and dissimilar from one class to another. For a classification in k classes, these algorithms generate an initial partition, and then search to improve it by reassigning individuals from one class to another, which allows the possibility that a poor initial partition could be corrected later.

In our context of designing an optimized CN-DW, we use Clarans (as a partition algorithm) in order to propose the best grouping of Rel-DW attributes in the targeted CN-DW columns families. Indeed, it makes a stochastic search based on different parameters allowing to limit the number of iterations of the method, as well as on random sampling. Given k the number of clusters sought, a data partitioning consists of a set of k medoids, to which are associated the set of objects according to their proximity to these medoids. The main steps of the method are as follows:

1. select a representative sample of data;
2. iterate a fixed number of times;

   - choose a random solution: a set of k medoid;

- iterate a fixed number of times:
  - choose a neighboring solution of the current one by a random modification of one of the solution's medoid;
  - keep the neighbor as the new current solution if the global inertia of the partition is less than the previous solution inertia;
- store the local optimal solution found

3. return the best of the local optimal solutions found.

CLARANS allows to extract classes of better quality compared to the PAM and CLARA methods; however this method is sensitive to the chosen parameters and has a complexity of the order $O(k.n^2)$.

4 PROPOSED APPROACH

In this section, we describe our novel contribution illustrated via the functional architecture presented in Figure 1. This architecture is composed of five principal layers:

Rel-DW layer: contains the data source of our system as a relational DW (Rel-DW) with its metadata and the most frequent decisional queries with their access frequencies.

Rel-DW to CN-DW layer: presents the main phase in our proposed solution that operates on the previous elements in order to group each set of attributes used together (in the selected input decisional queries) in the same column family to design an optimized targeted CN-DW, with Clarans clustering algorithm, that take into consideration all Rel-DW attributes in order to design a complete targeted model.

CN-DW layer: the logical optimized schema provided by the second layer is used to implements the CN-DW on HBase [23], considered as the most used NoSQL column oriented DB. This new system is fed from input source (Rel-DW) by respecting its logical schema.

Decision Maker & CN-DW Enrichment Layers: these layers correspond to the on-demand Big ETL that feed the CN-DW from external big data sources (Linked Open Data, Data Lakes, social media, semantic data, etc.) only when the decision makers need to enrich some answers of decisional queries judged as unsatisfactory to the needs of managers. In fact, this phase requires the adaptation of internal decision queries to external system schema in order to be able to extract perfectly the required data fragments. The external results are exploited in two ways:

1. they are integrated into CN-DW for future analyzes more fruitful;
2. they are merged with the results of internal decision query in order to be visualized by end users for more efficient managerial decisions.

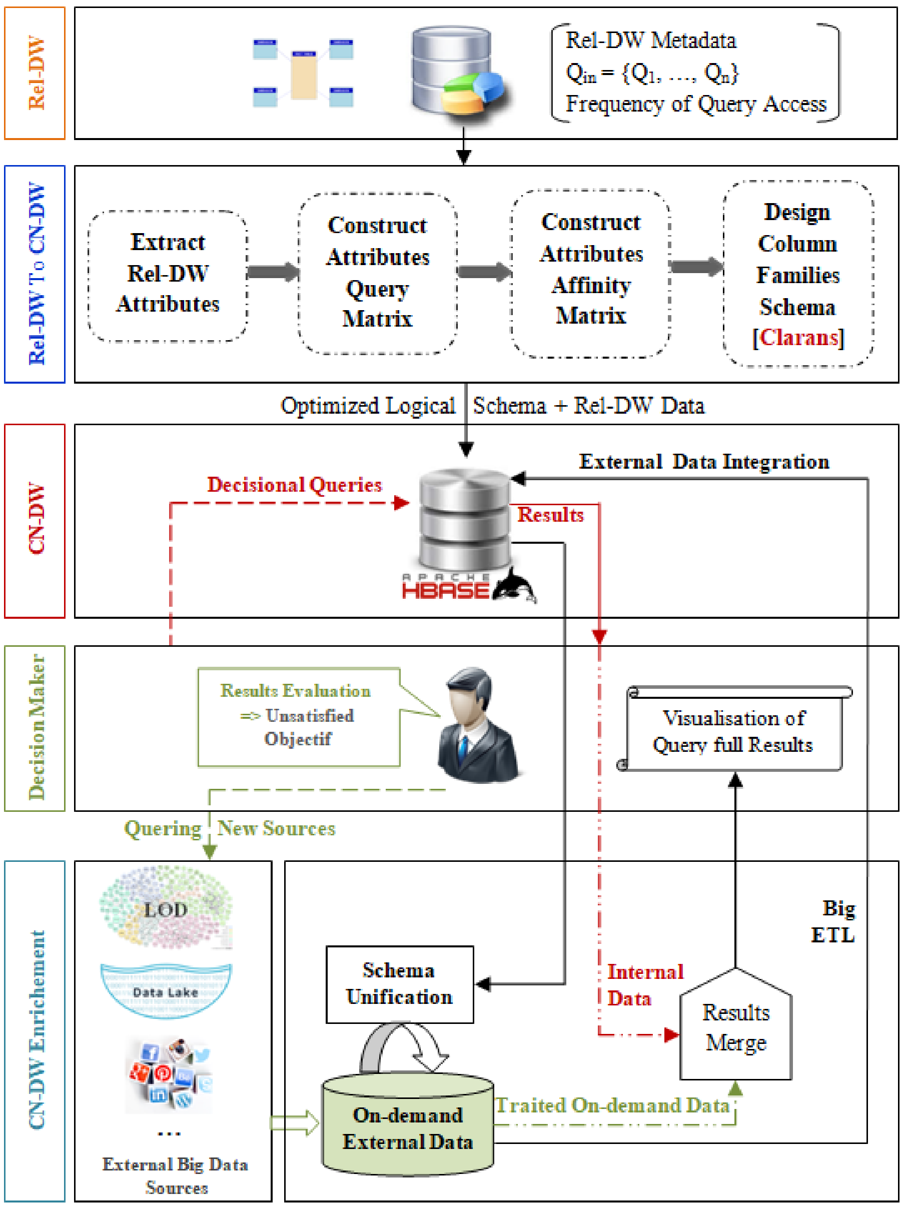

Rel-DW
Rel-DW Metadata
$Q_{in} = \{Q_1, \ldots, Q_n\}$
Frequency of Query Access
Rel-DW To CN-DW
Extract Rel-DW Attributes
Construct Attributes Query Matrix
Construct Attributes Affinity Matrix
Design Column Families Schema [Clarans]
Optimized Logical Schema + Rel-DW Data
CN-DW
Decisional Queries
External Data Integration
Results
APACHE HBASE
Decision Maker
Results Evaluation => Unsatisfied Objectif
Visualisation of Query full Results
Quering New Sources
CN-DW Enrichement
LOD
Data Lake
...
External Big Data Sources
Big ETL
Schema Unification
Internal Data
Results Merge
Traited On-demand Data
On-demand External Data

Figure 1. Global Architecture of our approach

In this paper, we will focus specially on the second layer designing an optimized CN-DW with Clarans clustering algorithm as presented in Algorithm-1. It operates on relational DW metadata (Rel-DW), list of most frequent decisional queries (Qdecisional), Frequency of Query Access Matrix by sites (FQAM), initial number of cluster defining column families (k), maximal number of neighbors (max_neighbor) and the iterations number of CLARANS clustering algorithm to resolve the problem (iterationsNbr) in order to design an optimized model for the targeted DW based on NoSQL column-oriented DB (CN-DW) that we represent as a big column NoSQL table (TCN-DW). In fact, our approach processes according to five major steps as described in Algorithm-1:

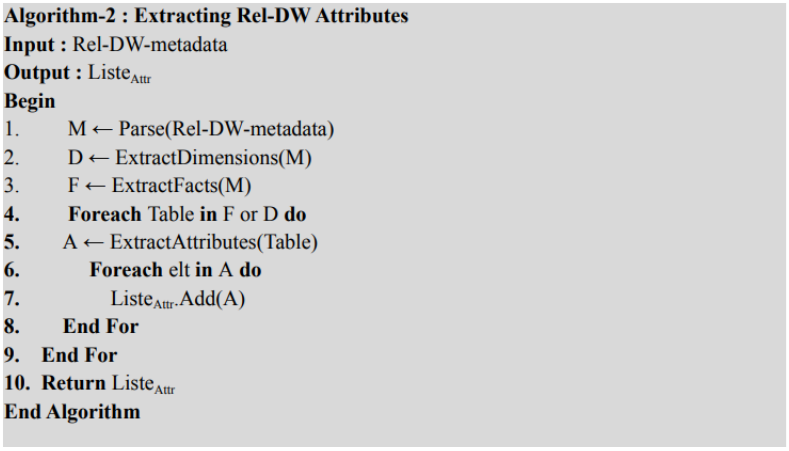
- Step 1: Extract the set of attributes
- Step 2: Construct Attributes Query Matrix (AQM)
- Step 3: Construct Attributes Affinity Matrix (AAM)
- Step 4: Design column family schemas (with Clarans algorithm)
- Step 5: Create final column-oriented NoSQL table (CN-DW)

```
Algorithm-1 : Rel-DW to CN-DW
Inputs : Rel-DW-metadata, List $Q_{decisional}$, FQAM, k, max_neighbor, iterationsNbr
Output : $T_{CN-DW}$
Begin
  List $L_{attr}$, AQM, AAM
  List<table[]>  CFs<Attributes>
1. $L_{attr}$ ← ExtractAttributes(Rel-DW, List $Q_{decisional}$)
2. AQM ← ConstructAttributesQueryMatrix($L_{attr}$, $Q_{decisional}$)
3. nbrQ ← $Q_{decisional}$.lenght() //number of queries
4. nbrAtt ←A.lenght()  //number of attributes
5. AAM ← ConstructAttributesAffinityMatrix(AQM, FQAM, nbrQ, nbrAtt)
6. CFs ← Clarans(AAM, k, max_neighbor, iterationsNbr)
7. $T_{CN-DW}$ ← CreateOrientedColumnTable(table_name, CFs)
8. Return $T_{CN-DW}$
End Algorithm
```

Figure 2. Algorithm-1

4.1 Extract Rel-DW Attributes

We aim in this step to prepare the main object in our major process to design column families in the targeted DW. In fact, and as illustrated in Algorithm-2, we operate on Rel-DW metadata to access its dimensions and facts in order to return all Rel-DW attributes that will be used by the next step.



```
Algorithm-2 : Extracting Rel-DW Attributes
Input : Rel-DW-metadata
Output : $Liste_{Attr}$
Begin
1.      M ← Parse(Rel-DW-metadata)
2.      D ← ExtractDimensions(M)
3.      F ← ExtractFacts(M)
4.      Foreach Table in F or D do
5.      A ← ExtractAttributes(Table)
6.          Foreach elt in A do
7.              $Liste_{Attr}$.Add(A)
8.      End For
9.   End For
10.  Return $Liste_{Attr}$
End Algorithm
```

Figure 3. Algorithm-2

4.2 Construct Attributes Query Matrix

In order to design CN-DW columns by grouping a set of attributes used together (in decisional queries) in the same column family, we have to conceive firstly the Attributes Query Matrix (AQM) as illustrated in Algorithm-3. It operates on the previous list of Rel-DW attributes $\mathit{ListeAttr}\{a_1, \ldots, a_n\}$ (generated by Algorithm-2) and a list of the most frequent decisional queries $QOLAP = \{q_1, \ldots, q_m\}$ presented as inputs in order to construct the AQM dimensioned by $n * m$ where n is the number of $QOLAP$ and m is the number of attributes. In fact, we go throught the previous lists to check the membership of each attribute to the set of queries: if an attribute a_i appears in q_j then the $AQM[i, j]$ is equal to 1 else $AQM[i, j]$ is equal to 0.

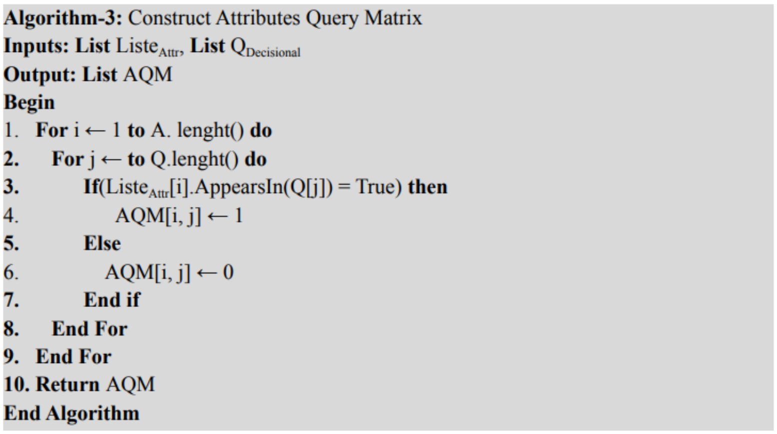
4.3 Construct Attributes Affinity Matrix

AAM is a symmetric matrix ($n \times n$ where n is the number of attributes) that indicates how the attributes are closely related. Each element of the AAM matrix is defined by an affinity value which measures the strength of an imaginary link between two attributes based on the fact that these attributes are used together by the query. The affinity value between two attributes a_i and a_j represents the number of times where two attributes are accessed together on all sites, it is defined as follows:

$$\mathit{Aff}(a_i, a_j) = \Sigma_{\text{All queries that access } a_i \text{ and } a_j} \textbf{Query access}, \tag{1}$$

where

$$\textbf{Query access} = \Sigma_{\text{For all sites}} \textbf{Access frequency of query}. \tag{2}$$



```
Algorithm-3: Construct Attributes Query Matrix
Inputs: List $Liste_{Attr}$, List $Q_{Decisional}$
Output: List AQM
Begin
1.  For i ← 1 to A. lenght() do
2.    For j ← to Q.lenght() do
3.      If($Liste_{Attr}$[i].AppearsIn(Q[j]) = True) then
4.        AQM[i, j] ← 1
5.      Else
6.        AQM[i, j] ← 0
7.      End if
8.    End For
9.  End For
10. Return AQM
End Algorithm
```

Figure 4. Algorithm-3

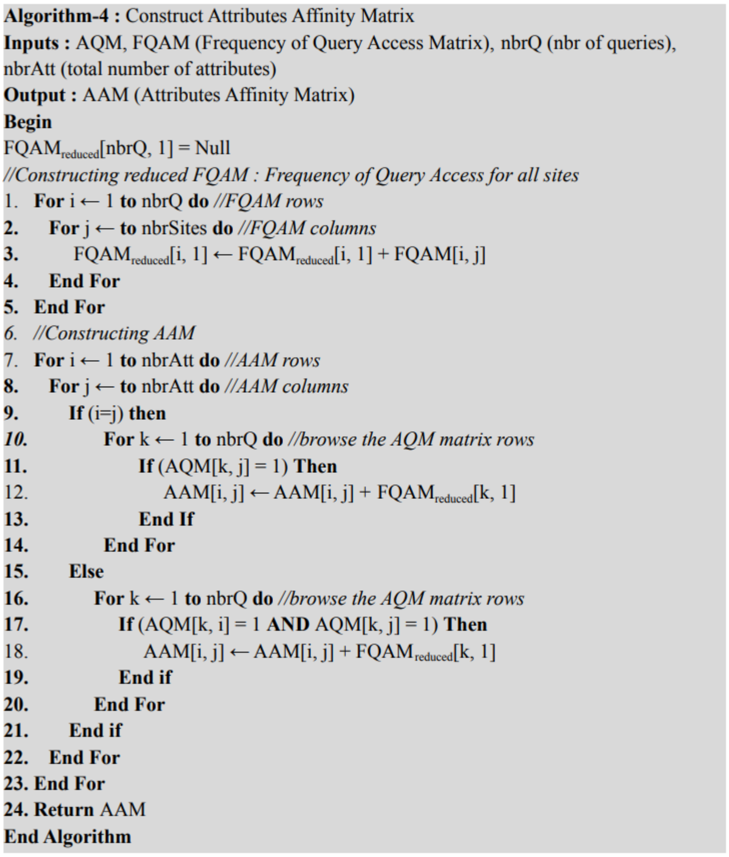
The Algorithm-4 below describes the detailed conception steps of the AAM matrix. It takes as inputs AQM generated by Algorithm-3, Frequency of Query Access Matrix, queries number and the total number of attributes in Rel-DW in order to design Attributes Affinity Matrix (AAM) by implementing the previous equations.

4.4 Design Column Family Schemas with Clarans

In order to overcome the limitations of using k-means and k-medoid for designing a CN-DW in existing approaches as detailed in the previous section, we opted in our solution for Clarans algorithm. In fact, CLARANS (*Clustering large Application Based on Randomized Search*) is a partitioning method for clustering a large database [20]. It has proven its effectiveness in generating an optimal number of clusters unlike the first two algorithms keeping the same number initially proposed without any improvement. The Algorithm-5 describes the main conception steps adopted by Clarans to design an optimal grouping of CN-DW column families.

4.5 Create Column Oriented Table

After generating the schema of columns families (CFs) by Clarans clustering algorithm (Algorithm-5), the Algorithm-6 (as described below) operates on these CFs to create an HBase column-oriented table which will be ready to receive and group optimally the data from Rel-DW.



```
Algorithm-4 : Construct Attributes Affinity Matrix
Inputs : AQM, FQAM (Frequency of Query Access Matrix), nbrQ (nbr of queries),
nbrAtt (total number of attributes)
Output : AAM (Attributes Affinity Matrix)
Begin
$FQAM_{reduced}$[nbrQ, 1] = Null
//Constructing reduced FQAM : Frequency of Query Access for all sites
1.  For i ← 1 to nbrQ do //FQAM rows
2.    For j ← to nbrSites do //FQAM columns
3.       $FQAM_{reduced}$[i, 1] ← $FQAM_{reduced}$[i, 1] + FQAM[i, j]
4.    End For
5.  End For
6.  //Constructing AAM
7.  For i ← 1 to nbrAtt do //AAM rows
8.    For j ← to nbrAtt do //AAM columns
9.      If (i=j) then
10.         For k ← 1 to nbrQ do //browse the AQM matrix rows
11.             If (AQM[k, j] = 1) Then
12.                AAM[i, j] ← AAM[i, j] + $FQAM_{reduced}$[k, 1]
13.             End If
14.         End For
15.     Else
16.        For k ← 1 to nbrQ do //browse the AQM matrix rows
17.           If (AQM[k, i] = 1 AND AQM[k, j] = 1) Then
18.              AAM[i, j] ← AAM[i, j] + $FQAM_{reduced}$[k, 1]
19.           End if
20.        End For
21.     End if
22.   End For
23. End For
24. Return AAM
End Algorithm
```

Figure 5. Algorithm-4

5 EXPERIMENTS RESULTS

In order to evaluate the performance of our system, we have implemented three different methods with python language: two already existing approaches using k-means and k-medoid clustering algorithm in addition to our method operating with Clarans, for designing an optimized CN-DW HBase system. We have operated on TPC-DS benchmark1 as a dataset. It is a relational data warehouse as a constellation schema, but in our context, we extracted a start schema composed of the fact table store_sales with its dimensions (Customer, Customer demographics, Customer address, Item, Time, Date, Household demographics, Promotion, Store). This start schema produces 176 attributes. To carry out our experiments, we set up two storage environments: the first one is relational with Intel-Core machine TMi5-4210U CPU@1.70 GHZ with 8 GB of RAM, and a 250 GB disk; it runs under the Windows 10 operating system with 64-bit. The second is a distributed NoSQL storage

```
Algorithm-5 : Design Columns Families schemas with Clarans
Inputs : AAM, k, max_neighbor, iterationsNbr
Output : medoidsOptimal
Begin
1.  OptimalCost ← 1000000
2.  medoidsOptimal ← Null
3.  i ← 1
4.  Do
5.  CurrentMedoids ← ConstructColumnFamilies(AAM, k)
6.  j ← 1
7.  Do
8.       NewMedoids ← RandomNeighborMedoids(CurrentMedoids)
9.       If ($TotalCost_{NewMedoids}$ < $TotalCost_{CurrentMedoids}$) Then
10.              CurrentMedoids ← NewMedoids
11.      Else
12.              j ← j +1
13.      End If
14. While (j < max_neighbor)
15. If ($TotalCost_{CurrentMedoids}$ < OptimalCost) Then
16.      OptimalCost ← $TotalCost_{CurrentMedoids}$
17.      medoidsOptimal ← CurrentMedoids
18. End if
19. i ← i +1
20. While (i < iterationsNbr)
21. Return medoidsOptimal
End Algorithm
```

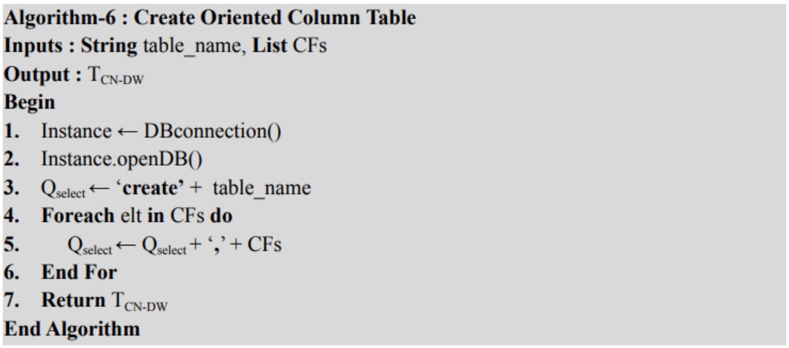
Figure 6. Algorithm-5

environment, as a Cloudera virtual machine version 5.13.0 with 7 GB of RAM and 2 CPU.

In order to measure the effectiveness and the quality of each method, we have fixed some evaluation factors to do this:

1. the number of column families generated,
2. the cost of the attribute groups schema and
3. the execution time.

The number of column families generated: As presented in Table 2, our proposed approach (Clarans CN-DW) optimizes the NbrCF by generating a lower number than the initial one from a NbrCF equals to 8, unlike the other existing methods (k-means and k-medoid CN-DW) that design CN-DW based on the initial NbrCF without any optimization. We observe that for NbrCF between 160 and 175, our method generates an optimal schema with final NbrCF equals to 45 and a Square Error (defined in the sub-section bellow) equals to zero.



```
Algorithm-6 : Create Oriented Column Table
Inputs : String table_name, List CFs
Output : $T_{CN-DW}$
Begin
1. Instance ← DBconnection()
2. Instance.openDB()
3. $Q_{select}$ ← 'create' + table_name
4. Foreach elt in CFs do
5.     $Q_{select}$ ← $Q_{select}$ + ',' + CFs
6. End For
7. Return $T_{CN-DW}$
End Algorithm
```

Figure 7. Algorithm-6

Initial NbrCF	2	8	10	25	30	40	160	168	170	172	174	175
Generated NbrCF	2	6	7	9	15	16	45	45	45	45	45	45

Table 2. Number of columns families generated by Clarans CN-DW method

The cost of the attributes groups schema: this metric is obtained using the Square Error (E2) as presented in [24]. In fact, the clustering method becomes more and more optimal as long as the E2 value approaches zero. The Square Error is calculated as follow:

$$E_S^2 = \sum_{i=1}^{k} \sum_{l=1}^{n} \left[(f_{q_l})^2 \times \alpha_i^{q_l} \left(1 - \frac{\alpha_i^{q_l}}{\beta_i} \right) \right] \tag{3}$$

with f_{q_l} is the access frequency of q_l query, αq_{l_i} is the number of attributes in CF_i accessed by q_l query, β_i is the number of attributes in column family CF_i.

In order to measure the quality of the attributes groups schema, we have calculated the Square Error of each method by varying the number of columns families (NbrCF). As presented in Table 3, the Square Error is equal to zero for K-means and K-medoid CN-DW methods from NbrCF = 45. However, the Square Error of Clarans CN-DW method is equal to zero from NbrCF = 160.

	K-Means	**K-Medoid**	**Clarans**
NbrCF	45	45	45
Square Error	0.0		

Table 3. Square Error for the three CN-DW methods

The execution time: the graph illustrated in Figure 2 describes the execution time variation for the three CN-DW methods (using K-means, K-medoid and Clarans clustering algorithm) by varying the number of columns families (NbrCF) between 2 and 45. In fact, we notice that the execution time of the

Clarans CN-DW method is very high compared to the other ones. However, as described in Figure 3, the execution time of the K-means CN-DW method is higher than K-medoid CN-DW one for $2 \leq \text{NbrCF} \leq 9$. On the other hand, for $10 \leq \text{NbrCF} \leq 45$ the execution time of K-medoid CN-DW method is very high compared to K-means one. Hence, we deduce the following arrangement of the three compared CN-DW methods: K-means < K-medoid < Clarans.

Table 4 presents a detailed execution time of generating CN-DW's CF number with K-means, K-medoid and Clanrans clustering algorithms.

		K-Means	**K-Medoid**	**Clarans**
Number of initial Columns families	2	0.062479	0.031249	13.790757
	10	0.12484	0.154433	8.250328
	25	0.289458	0.563852	23.487544
	30	0.285789	0.693265	17.856511
	40	0.364845	0.858601	23.44205
	45	0.409217	1.029528	26.946319

Table 4. Execution time of generating CN-DW's CF number with clustering algorithms

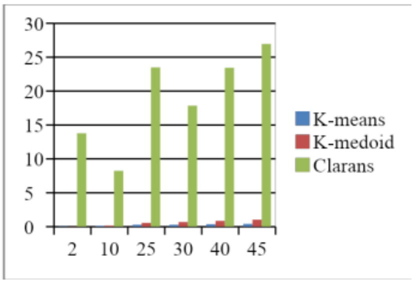


Figure 8. Execution time of generating CN-DW's CF number with K-means, K-medoid and Clarans

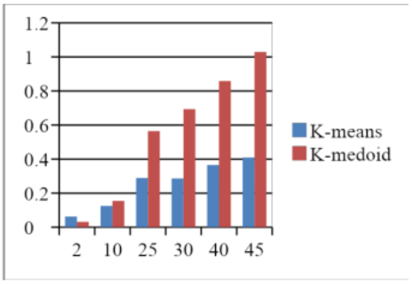


Figure 9. Comparison between K-means and K-medoid execution time

6 DISCUSSION

By analyzing the previous experiments results summarized in Table 5, we have noted that in the optimal case when the Square Error is equal to zero, the attributes not used by the input decisional queries are grouped in the same column family by our proposed method with Clarans, unlike the two existing methods which manipulate just the decisional queries attributes that conceive an incomplete CN-DW unsuitable for new needs. Although our approach has recorded a high execution time compared to existing methods, Table 2 proven its effectiveness in optimizing the number of column families unlike the two existing methods which preserves the same initial number and do not offer any improvement in this direction. In fact, our proposed approach always generates a number of columns families lower than the initial number especially in the case where this initial number is equal to (the number of attributes -1), our method returns an optimal schema in terms of execution time, number of clusters and Square Error. In addition, the execution time of our method depends on the number of iterations and the number of neighbors, if these parameters are small, the execution time is fast but the schema is not optimal, in the opposite case, the schema is optimal but the execution time is considerable.

	Initial CF Number	**Final CF Number**	**Execution Time**	**Square Error**
K-Means	45	45	0.405	0.0
K-Medoid	45	45	1.029	0.0
Clarans	175	45	20.480	0.0

Table 5. Summary of the main experiments results

7 CONCLUSION

In this contribution, we have proposed a new method to optimize designing columnar data warehouse using CLARANS clustering algorithm that generates an optimal grouping of column families. In addition, our system take into consideration all Rel-DW attributes to design a complete targeted model able to meet perfectly the new needs of business leaders. In order to improve the effectiveness and benefits of our system, we have elaborated numerous comparison tests (with existing works) based on TPC-DS[1] benchmark as a dataset. As future work, we intend to implement our CN-DW with the generated combination of columns families by CLARANS algorithm and execute the set of selected decisional queries on our optimized CN-DW to evaluate its performance.

[1] `http://www.tpc.org/tpcds/`

REFERENCES

[1] Chandra, P.—Gupta, M. K.: Comprehensive Survey on Data Warehousing Research. International Journal of Information Technology, Vol. 10, 2018, No. 2, pp. 217–224, doi: 10.1007/s41870-017-0067-y.

[2] Santos, M. Y.—Costa, C.—Galvão, J.—Andrade, C.—Pastor, O.—Marcén, A. C.: Enhancing Big Data Warehousing for Efficient, Integrated and Advanced Analytics. In: Cappiello, C., Ruiz, M. (Eds.): Information Systems Engineering in Responsible Information Systems (CAiSE 2019). Springer, Cham, Lecture Notes in Business Information Processing, Vol. 350, 2019, pp. 215–226, doi: 10.1007/978-3-030-21297-1_19.

[3] Ng, R. T.—Han, J.: CLARANS: A Method for Clustering Objects for Spatial Data Mining. IEEE Transactions on Knowledge and Data Engineering, Vol. 14, 2002, No. 5, pp. 1003–1016, doi: 10.1109/TKDE.2002.1033770.

[4] Nebot, V.—Berlanga, R.: Building Data Warehouses with Semantic Data. Proceedings of the 2010 EDBT/ICDT Workshops (EDBT '10), 2010, doi: 10.1145/1754239.1754250.

[5] Boumlik, A.—Soussi, N.—Bahaj, M.: SMART-ETL-MR: Novel ETL Framework for Building Data Warehouse from Big Data Source Using MapReduce. Journal of Theoretical and Applied Information Technology, Vol. 98, 2020, No. 17, pp. 3449–3460.

[6] Yangui, R.—Nabli, A.—Gargouri, F.: Towards Data Warehouse Schema Design from Social Networks – Dynamic Discovery of Multidimensional Concepts. Proceedings of the 17th International Conference on Enterprise Information Systems – Volume 2 (ICEIS 2015), 2015, pp. 338–345, doi: 10.5220/0005383903380345.

[7] Dehdouh, K.: Building OLAP Cubes from Columnar NoSQL Data Warehouses. In: Bellatreche, L., Pastor, Ó., Almendros Jiménez, J. M., Aït-Ameur, Y. (Eds.): Model and Data Engineering (MEDI 2016). Springer, Cham, Lecture Notes in Computer Science, Vol. 9893, 2016, pp. 166–179, doi: 10.1007/978-3-319-45547-1_14.

[8] Llave, M. R.: Data Lakes in Business Intelligence: Reporting from the Trenches. Procedia Computer Science, Vol. 138, 2018, pp. 516–524, doi: 10.1016/j.procs.2018.10.071.

[9] Berkani, N.—Bellatreche, L.—Khouri, S.—Ordonez, C.: Value-Driven Approach for Designing Extended Data Warehouses. In: Song, I. Y., Romero, O., Wrembel, R. (Eds.): Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP 2019). CEUR Workshop Proceedings, Vol. 2324, 2019.

[10] Khouri, S.—Berkani, N.—Bellatreche, L.—Lanasri, D.: Data Cube Is Dead, Long Life to Data Cube in the Age of Web Data. In: Madria, S., Fournier-Viger, P., Chaudhary, S., Reddy, P. K. (Eds.): Big Data Analytics (BDA 2019). Springer, Cham, Lecture Notes in Computer Science, Vol. 11932, 2019, pp. 44–64, doi: 10.1007/978-3-030-37188-3_4.

[11] Berkani, N.—Bellatreche, L.—Khouri, S.—Ordonez, C.: The Contribution of Linked Open Data to Augment a Traditional Data Warehouse. Journal of Intelligent Information Systems, Vol. 55, 2020, No. 3, pp. 397–421, doi: 10.1007/s10844-020-00594-w.

[12] Yangui, R.—Nabli, A.—Gargouri, F.: Automatic Transformation of Data Warehouse Schema to NoSQL Data Base: Comparative Study. Procedia Computer Science, Vol. 96, 2016, pp. 255–264, doi: 10.1016/j.procs.2016.08.138.

[13] Chevalier, M.—Malki, M. E.—Kopliku, A.—Teste, O.—Tournier, R.: Implementation of Multidimensional Databases in Column-Oriented NoSQL Systems. In: Morzy, T., Valduriez, P., Bellatreche, L. (Eds.): Advances in Databases and Information Systems (ADBIS 2015). Springer, Cham, Lecture Notes in Computer Science, Vol. 9282, 2015, pp. 79–91, doi: 10.1007/978-3-319-23135-8_6.

[14] Dehdouh, K.—Bentayeb, F.—Boussaid, O.—Kabachi, N.: Using the Column Oriented NoSQL Model for Implementing Big Data Warehouses. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '15), The Steering Committee of the World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015, pp. 469–475.

[15] Prakash, D.: NOSOLAP: Moving from Data Warehouse Requirements to NoSQL Databases. Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2019), 2019, pp. 452–458, doi: 10.5220/0007748304520458.

[16] Scabora, L. C.—Brito, J. J.—Ciferri, R. R.—de Aguiar Ciferri, C. D.: Physical Data Warehouse Design on NoSQL Databases – OLAP Query Processing over HBase. Proceedings of the 18th International Conference on Enterprise Information Systems – Volume 1 (ICEIS), SciTePress, 2016, pp. 111–118, doi: 10.5220/0005815901110118.

[17] Mior, M. J.—Salem, K.—Aboulnaga, A.—Liu, R.: NoSE: Schema Design for NoSQL Applications. IEEE Transactions on Knowledge and Data Engineering, Vol. 29, 2017, No. 10, pp. 2275–2289, doi: 10.1109/TKDE.2017.2722412.

[18] Boussahoua, M.—Boussaid, O.—Bentayeb, F.: Logical Schema for Data Warehouse on Column-Oriented NoSQL Databases. In: Benslimane, D., Damiani, E., Grosky, W. I., Hameurlain, A., Sheth, A., Wagner, R. R. (Eds.): Database and Expert Systems Applications (DEXA 2017). Springer, Cham, Lecture Notes in Computer Science, Vol. 10439, 2017, pp. 247–256, doi: 10.1007/978-3-319-64471-4_20.

[19] Boussahoua, M.—Bentayeb, F.—Boussaid, O.—Kabachi, N.: A Data Partitioning Optimization Approach for Distributed Data Warehouses on Column Family NoSQL Systems. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '18), The Steering Committee of the World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2018, pp. 54–60.

[20] Soni, K. G.—Patel, A.: Comparative Analysis of K-Means and K-Medoids Algorithm on IRIS Data. International Journal of Computational Intelligence Research, Vol. 13, 2017, No. 5, pp. 899–906.

[21] Boumlik, A.—Soussi, N.—Bahaj, M.: Automatic Data Modeling Transformation Approach of NoSQL Document and Column Stores to RDF. Journal of Theoretical and Applied Information Technology, Vol. 96, 2018, No. 15.

[22] Verma, R.—Puntambekar, D. M.: Comparison of Partitioning Algorithms for Categorical Data in Cluster. International Journal of Engineering Science and Com-

puting, Vol. 8, 2018, No. 7, Art. No. 18701.

[23] Strauch, C.: NoSQL Databases. Lecture Notes, Stuttgart Media University, Vol. 20, 2011, No. 24, pp. 1–149.

[24] Derrar, H.—Boussaid, O.—Ahmed-Nacer, M.: An Objective Function for Evaluation of Fragmentation Schema in Data Warehouse. Encyclopedia of Information Science and Technology, Third Edition, IGI Global, 2015, pp. 1949–1957, doi: 10.4018/978-1-4666-5888-2.ch188.

Nassima Soussi is an Assistant Professor in the Department of Mathematics and Computer Sciences from National School of Applied Sciences, Sultan Moulay Slimane University (Khouribga, Morocco). She obtained her special higher studies degree in software engineering from the National School of Applied Sciences (Khouribga, Morocco) in 2014 and Ph.D. from the Faculty of Sciences and Technology, Hassan 1st University (Settat, Morocco) in 2018. Her main research domains are semantic web and big data warehousing.