# NOSQL IMPLEMENTATION OF A CONCEPTUAL DATA MODEL: UML CLASS DIAGRAM TO A DOCUMENT-ORIENTED MODEL

A.BENMAKHLOUF

Computer, Networks, Mobility and Modeling Laboratory (IR2M), Faculty of Science and Technology, University Hassan 1st, BP 577, 26000 Settat, Morocco

## ABSTRACT

*The relational databases have shown their limits to the exponential increase in the volume of manipulated and processed data. New NoSQL solutions have been developed to manage big data. These approaches are an interesting way to build no-relational databases that can support large amounts of data. In this work, we use conceptual data modeling (CDM), based on UML class diagrams, to create a logical structure of a NoSQL database, taking account the relationships and constraints that determine how data can be stored and accessible. The NoSQL logical data model obtained is based on the Document-Oriented Model (DOM). to eliminate joins, a total and structured nesting is done on the collections of the document-oriented database.*

*Rules of passage from the CDM to the Logical Oriented-Document Model (LODM) are also proposed in this paper to transform the different types of associations between class. An application example of this NoSQL BDD design method is realised to the case of an organization working in the e-commerce business sector.*

## KEYWORDS

*big data, No-Relational, conceptual data modelling, The NoSQL logical data model, nested document-oriented model.*

## 1. INTRODUCTION

In recent years data management are facing the challenges of 4V. Large Volume of Data, Variety of Data Types, Increasing Velocity, and Need to Manage Data Variability [1]. The relational model, which is at the root of most management applications, has shown its limits to these needs. So, the world has entered a new era of Bigdata database [2].The latter represents a set of new techniques in modelling, making it possible to move to a more advanced scale in terms of volume and type of data and the velocity of the processing of these data. In this context, NoSQL databases offer new storage solutions in large-scale environments, replacing the various traditional database management systems that are mostly relational.

The scope of NoSQL databases remains limited to simple data models. Some research has proposed NoSQL models for Datawarehouse [3][4][5]. Authors have also proposed NoSQL models for transactional databases [6] based on a normalised conceptual data model. Logical models of NoSQL data obtained introduce joins that make the data processing always very heavy. In this work we develop a NoSQL Logic model that is based on the Document-Oriented Model (DOM). In order to remove joins, the proposed model will be based on nesting collections while maintaining data consistency. The NoSQL logic model is obtained from a Conceptual Data Model that will be represented by the class diagram. Rules of passage will be proposed in this paper to transform a CDM into a DOM.

The objective of the CDM is to model data in a structured and consistent way to preserve information about management rules. It is an abstract representation of reality that will allow us to obtain a database that will be managed by a database management system (DBMS). In the classic case, this database will be relational, managed by an RDBMS. The latter applies the principles of relational mathematics to define, manipulate, query and secure data. The formalism of Relational Algebra is at the heart of the SQL query language used in RDBMS. Pass rules are well defined to switch from a CDM to a Relational Model (RM).

The CDM can also be applied to design a no-relational database. These DBs can be based on document, column or graph-oriented models. A study of the transition methods from a CDM to a NoSQL model DOM will be proposed in this article. The NoSQL database will be managed by the SMDB MongoDB. These transfer methods will be applied to a CDM that models an e-commerce activity.

## 2. DOCUMENT-ORIENTED MODEL

In this model each key is associated with a document. The documents are grouped in collections. Each document has a set of attribute values that can be atomic or sub-documented. The schema of the NOSQL database can be deduced from the conceptual model of data established for the relational model.

The collection in a DB is defined by a set of documents: $C^D = \{D_1, D_2, \ldots., D_n\}$

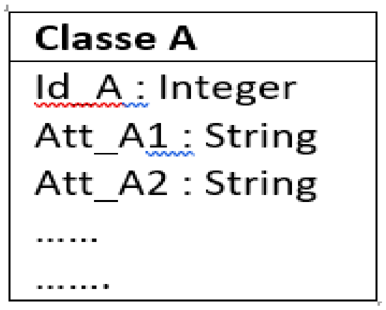
Each document $D_i$ is defined by a set of couples : $D_i = \{(A_1^i, V_1^{Di}), \ldots.., (A_{mi}^i, V_{mi}^{Di})\}$

- $A_j^{Di}$ is affected attribute to document Di
- $V_1^{Di}$ is a value of the attribute $A_j^{Di}$. This value can be:
    - Atomic value.
    - A nested sub-document consisting of a set of pairs (attribute, value).

## 3. THE RULES FOR PASSAGE FROM A CDM TO LDOM

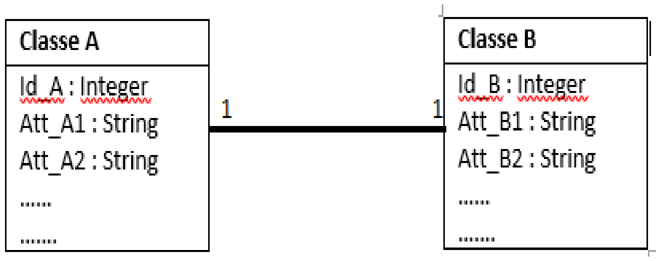
We present in this section the rules for passage from a class diagram to a NOSQL document-oriented model.

### 3.1. CLASS WITH ATTRIBUTES



each class is transformed into a collection consisting of a set of documents. Class attributes are transformed into attributes in each document.
$C^{classA} = \{Id\_A, Att\_A1, Att\_A2, \ldots..\}$

## 3.2. Association One To One



### 3.2.1. Document-Oriented Model with join

It will consist of two collections CClassA and CClassB with a migration of the primary key of one of the two collections to become a foreign key in the other collection.

$C^{ClassA} = \{id_A, Att_{A1}, Att_{A2}, \ldots\ldots, id_B\}$
$C^{ClassB} = \{id_B, Att_{B1}, att_{B2}, \ldots\ldots\}$

### 3.2.2. Nested Document-Oriented model

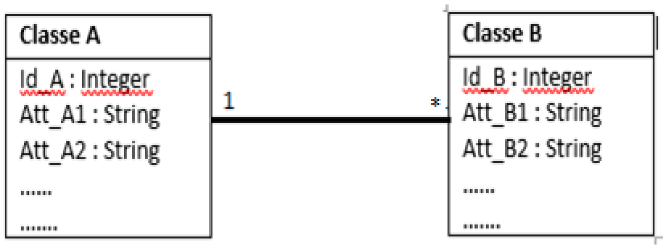
Creating a principal nested collection $C^P$ representing one of the two classes with these attributes. The other class becomes a sub-collection of the main collection.

$$C^{ClassA} = \left\{ \begin{matrix} id_A, Att_{A1}, Att_{A2}, \ldots\ldots, id_B, \\ N^{ClassB}: \{id_B, Att_{B1}, att_{B2}, \ldots\ldots\} \end{matrix} \right\}$$

The following JSON query gets the Nested Model from the joined model.

```
var result = db.CClassA.aggregate([{
     $lookup: {
         from: " CClassB",
         localField: "id_B",
         foreignField: " id_B",
         as: "CClassB"}}]);
  db.CP.insert(result.toArray());
```

## 3.3. Association one to many



### 3.3.1 Document-Oriented Model with join

Same rule of passage as the preceding case except that now it is inevitably the collection of the side several which receives as foreign key the primary key of the collection of the side one.

$C^{ClassB} = \{id_B, Att_{B1}, Att_{B2}, \ldots\ldots, id_A\}$
$C^{ClassA} = \{id_A, Att_{A1}, att_{A2}, \ldots\ldots\}$

### 3.3.2 Nested Document-Oriented model

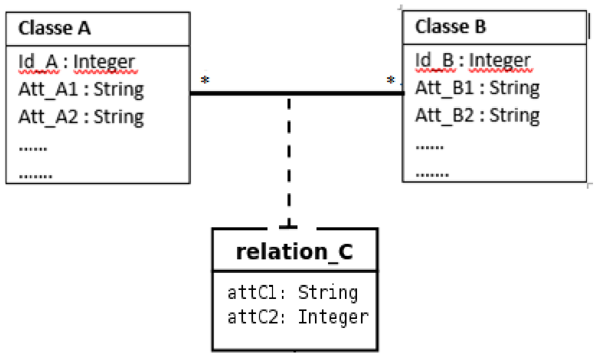
Creation of a new principal collection CP representing the class of sides several with these attributes. The class of side 1 becomes a sub-collection of the principal collection.

$$C^P = \begin{Bmatrix} id_B, Att_{B1}, Att_{B2}, \ldots\ldots, id_A, \\ N^{ClassA}: \{id_A, Att_{A1}, att_{A2}, \ldots\ldots\} \end{Bmatrix}$$

The following JSON query gets the Nested Model from the joined model.

```
var result = db.CClassB.aggregate([{
    $lookup: {
        from: " CClassA",
        localField: "id_A",
        foreignField: " id_A",
        as: "CClassA"}}]);
  db.CP.insert(result.toArray());
```

### 3.4. Association many to many



### 3.4.1. Document-Oriented Model with join

Creation of a new collection composed by the attributes of the association class and the primary keys of the classes that participated in the association. The database will therefore consist of the three collections:

$$C^{ClassB} = \{id_B, Att_{B1}, Att_{B2}, \ldots.\}$$
$$C^{ClassA} = \{id_A, Att_{A1}, att_{A2}, \ldots..\}$$
$$C^{ClassAB} = \{id_A, id_B, att_{C1}, att_{C2}\}$$

### 3.4.2. Nested Document-Oriented model

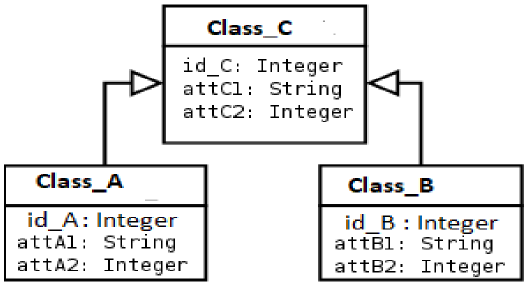
Creation of a main collection whose attributes are composed by those of the association class and by the primary keys of the classes that participated in the association. These are transformed into sub-collections containing their attributes.

$$C^{AB} = \begin{Bmatrix} id_A, id_B, att_{C1}, att_{C2}, \\ N^{ClassA}: \{id_A, Att_{A1}, att_{A2}, \ldots..\}, \\ N^{ClassB}: \{id_b, Att_{B1}, att_{B2}, \ldots..\} \end{Bmatrix}$$

The following JSON query gets the Nested Model from the joined model.
var result = db.C$^{ClassAB}$.aggregate([{

```
   $lookup: {
      from: " ClassA_COLLETION",
      localField: "id_A",
      foreignField: " id_A",
      as: "ClassA"},
  $lookup: {
      from: " ClassB_COLLETION",
      localField: "id_B",
      foreignField: " id_B",
      as: "ClassB"}}]);
db.C^AB.insert(result.toArray());
```

## 3.5. HERITAGE ASSOCIATION



### 3.5.1. Document-Oriented Model with join

The document-oriented model will be composed by the parent and son collections. In son collection we find the no-key attributes of this class plus a primary key composed of the keys of the parent and son classes. An attribute "type" will be added in the parent class. This attribute makes it possible to specify with which son class should be complete the information of a document of the collection of the parent class.

$C^{ClassC} = \{id_C, Att_{C1}, Att_{C2}, type \ldots.\}$
$C^{ClassA} = \{id_A, id_C, Att_{A1}, att_{A2}, \ldots..\}$
$C^{ClassB} = \{id_B, id_C, Att_{B1}, att_{B2}, \ldots\}$

### 3.5.2. Nested Document-Oriented model

Creating a collection that represents the parent class consisting of its primary key and its attributes plus the primary key of the son class.

$$C^P = \left\{ \begin{array}{c} id_C, att_{C1}, att_{C2}, type, \quad : \\ N^{ClassA}: \{id_A, id_{C,} Att_{A1}, att_{A2}, \ldots..\}, \\ N^{ClassB}: \{id_B, id_C, Att_{B1}, att_{B2}, \ldots..\} \end{array} \right\}$$

The following JSON query gets the Nested Model from the joined model.

```
var result = db.C^ClassC.aggregate([{
   $lookup: {from: " ClassA ",
      localField: "id_C",
      foreignField: " id_C",
      as: "ClassA"}
  $lookup: {from: " ClassB ",
      localField: "id_C",
      foreignField: " id_C",
      as: "ClassB"}}]);
  db.C^P.insert(result.toArray());
```

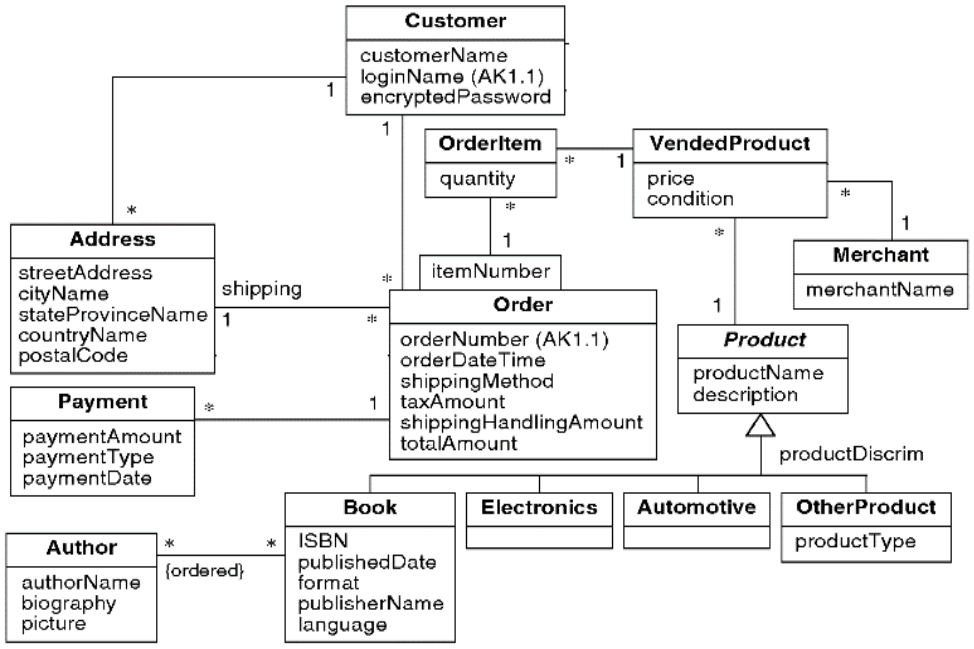
## 4. EXAMPLE OF CREATION OF A DOM FROM CDM (CLASS DIAGRAM).

The conceptual data model (CDM) used as an example to obtain a document-oriented model is given in Figure-1. It is a part of a class diagram that models the data of an e-commerce activity [6].

A customer can make several orders and an order belongs to a single customer. An order will be delivered to an address and several orders can be delivered to the same address. multiple customers may be assigned to a residential address. the payment of an order can be carried out in several payments which can be by a credit card, a check or money.

An order can consist of several OrderItems, each identified by an itemNumber. Each OrderItem is characterised by a quantity sold. An order item refers to a VendedProduct that is a product sold by a particular merchant. The price of a product may vary depending on the merchant and its condition as new, good condition and worn. Products for sales is three kinds: Book, Electronics, and Automotive.

Figure 1: Conceptual Data Model

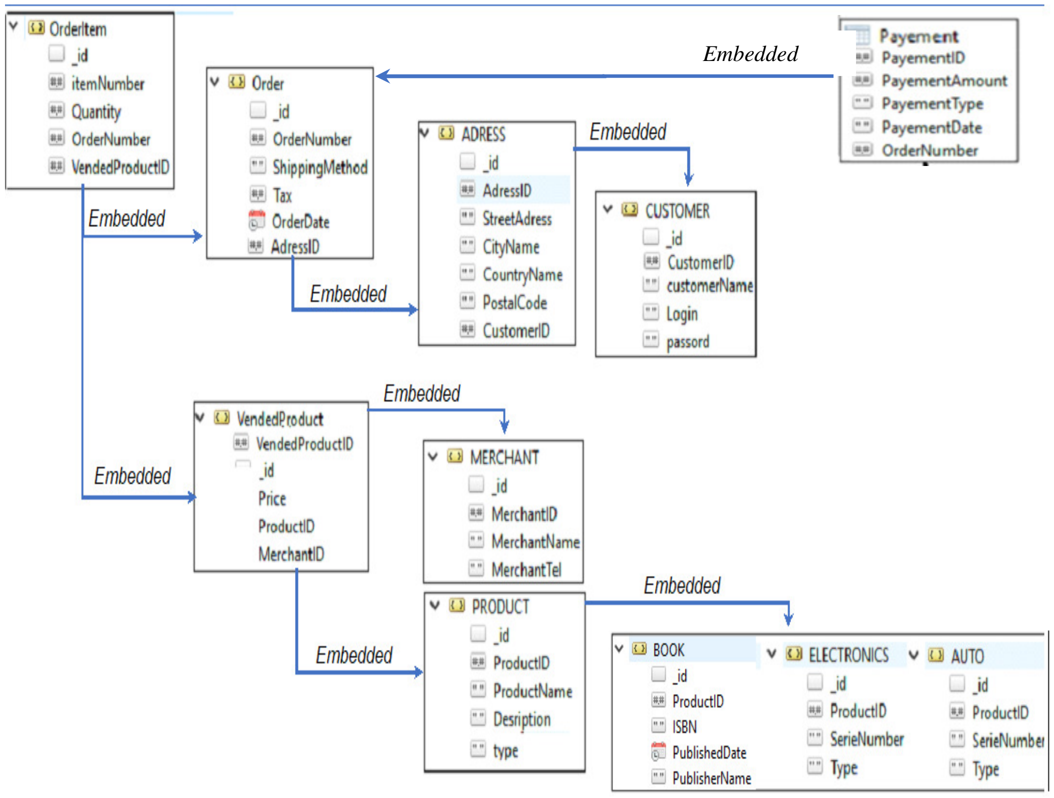


By applying the transfer rules mentioned above, we obtain the document-oriented database model. Since the latter is a nested model we represent the DOM by a hierarchical tree (figure-2). In this figure we can see the nesting of the collections that translate the different classes of the CDM. Class can be transformed collection, set of Attribute can be mapped to attribute in collection. The associations are transformed into joined collections and then into a nested collection model using the JSON requests mentioned above. in table-1 we present the correspondence between the elements of a class diagram and the nested DOM.

Table-1: correspondence between the elements of a class diagram and the nested DOM.

| UML Class Diagram | Logical Data Model Oriented-Document |
|---|---|
| Class | Collection |
| Attribute in class | Attribute in Collection |
| association | Embedded Collection |

Figure-2: Logical document-oriented model



## 4.1. APPLICATION OF QUERIES

To validate our embedded DOM modeling, we propose to compute analysis queries in the two database models: NoSQL Document-oriented and Relational SQL. PHP/MongoDB and PHP/MySQL connections are made to structure the query results in tables, especially the query results documents applied to the MongoDB database. These queries involve several analysis dimensions in order to test the model in the case of highly aggregated data calculations.

**Query-1:** Calculates the sales performance by each customer. In the MongoDB DB, this query is applied to the collection "OrderItem". It is given by the following JSON code:

```
db.OrderItem.aggregate([{ $group: { _id: {Code:"$Order.ADRESS.CUSTOMER.
CustomerID",
Nom : "$Order.ADRESS.CUSTOMER.
customerName"} ,
total: { $sum: {$multiply:["$VendedProduct.Price" ,"$Quantity"]}}}}}])
```

**Query-2:** This query calculates the sales performance by year and by country. The JSON code is:

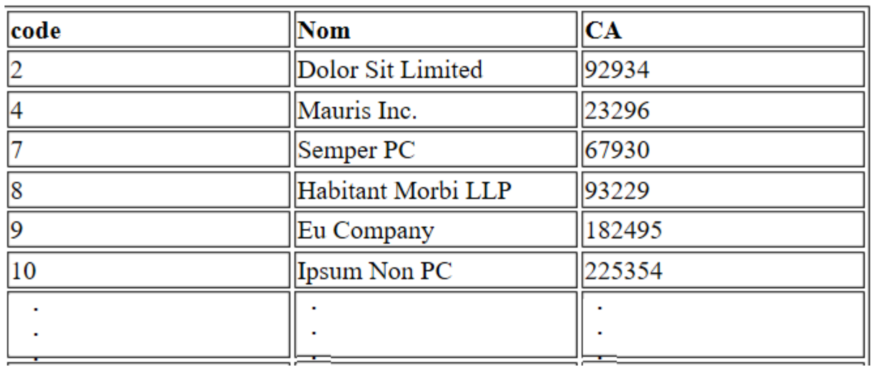
```
db.OrderItem.aggregate([{ $group: { _id: {
COUNTRY:"$Order.ADRESS.CountryName",
ANN:{$year: "$Order.OrderDate"}},
total:{$sum:{$multiply:["$VendedProduct.Price" ,"$Quantity"]}}}}])
```

**Query-3:** This query calculates the turnover by country, product and year. The JSON code is:

```
db.OrderItem.aggregate([{ $group: { _id: {
COUNTRY:"$Order.ADRESS.CountryName",
PRODUIT:"$VendedProduct.PRODUCT.ProductName", ANN:{$year: "$Order.OrderDate"}},
total:{$sum:{$multiply:["$VendedProduct.Price" ,"$Quantity"]}}}}])
```

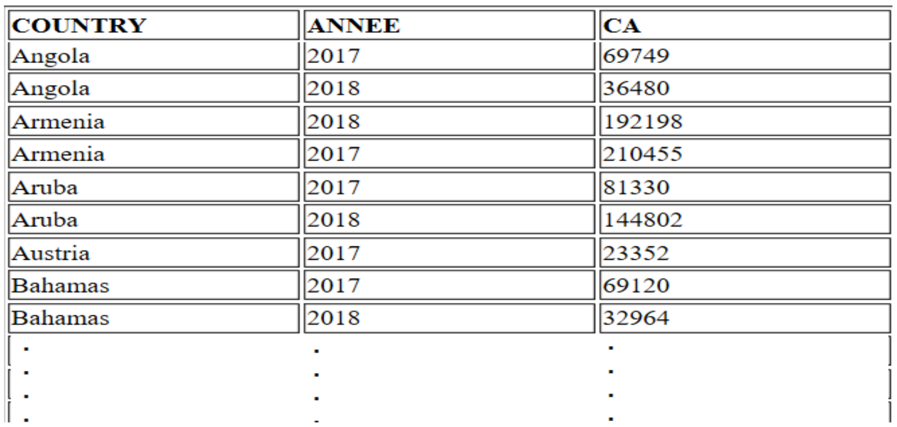
Result of the queries: In the tables 1, 2 and 3 Below, we present the results of the three queries applied simultaneously in the MongoDB and MySQL databases.

Table-1: Result Of The Query-1



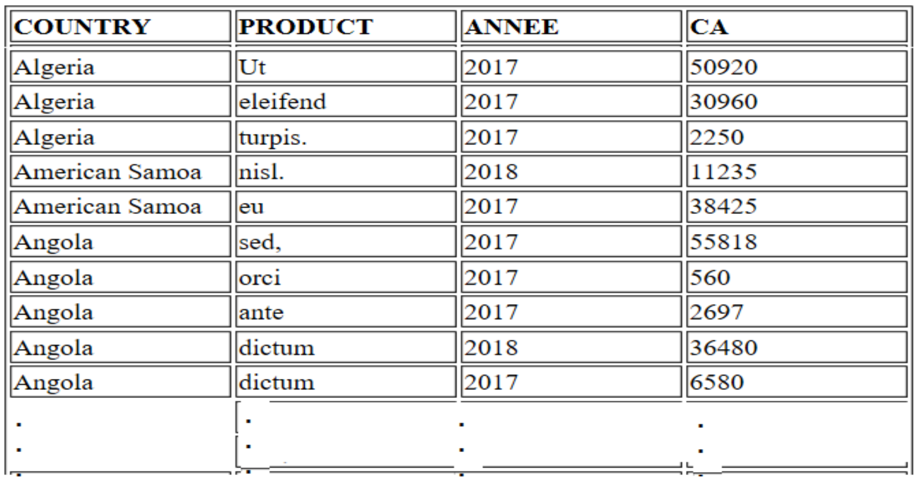
| code | Nom | CA |
|---|---|---|
| 2 | Dolor Sit Limited | 92934 |
| 4 | Mauris Inc. | 23296 |
| 7 | Semper PC | 67930 |
| 8 | Habitant Morbi LLP | 93229 |
| 9 | Eu Company | 182495 |
| 10 | Ipsum Non PC | 225354 |
| . | . | . |

Table-2: Result of the query-2



| COUNTRY | ANNEE | CA |
|---|---|---|
| Angola | 2017 | 69749 |
| Angola | 2018 | 36480 |
| Armenia | 2018 | 192198 |
| Armenia | 2017 | 210455 |
| Aruba | 2017 | 81330 |
| Aruba | 2018 | 144802 |
| Austria | 2017 | 23352 |
| Bahamas | 2017 | 69120 |
| Bahamas | 2018 | 32964 |
| . | . | . |

Table-3: Result of the query-3



| COUNTRY | PRODUCT | ANNEE | CA |
|---|---|---|---|
| Algeria | Ut | 2017 | 50920 |
| Algeria | eleifend | 2017 | 30960 |
| Algeria | turpis. | 2017 | 2250 |
| American Samoa | nisl. | 2018 | 11235 |
| American Samoa | eu | 2017 | 38425 |
| Angola | sed, | 2017 | 55818 |
| Angola | orci | 2017 | 560 |
| Angola | ante | 2017 | 2697 |
| Angola | dictum | 2018 | 36480 |
| Angola | dictum | 2017 | 6580 |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |

## 5. CONCLUSION

The interest of this work is to develop a method of no-relational modelling of a production database. The proposed NoSQL model is obtained from a conceptual data modelling previously performed based on the standard UML modelling language. The no-relational model obtained is based on a structured nesting of the document collections which will enable us, on the one hand, to preserve the coherence of the data and on the other hand to eliminate the joins between the collections. This will, of course, considerably reduce the response time of multidimensional analysis requests even in the case of a large amount of data. Nesting of collections is easily achieved using JSON aggregation queries applied on a standardized document-oriented template with join.

In this article we have developed only the transition phase of the Conceptual Data Model to the NoSQL logic model. The physical data model is implemented in the Mongodb DBMS. different requests for analysis have been made successfully. the study of performance by comparing of the response times of multidimensional aggregation queries will be developed in the next article.

## REFERENCES

[1] A B M Moniruzzaman and Syed Akhter Hossain, 2013, "NoSQL Database: New Era of Databases for Big-data Analytics - Classification, Characteristics and Comparison,". International Journal of Database Theory and Application.

[2] Veronika Abramova, Jorge Bernardino and Pedro Furtado, 2014, "Experimental Evaluation Of NOSQL DataBase", International Journal of Database Management Systems (IJDMS) Vol.6, No.3, June 2014

[3] Y.Hiyane, A.Benmakhlouf, A.Marzouk, 2018, "Storing data in NOSQL data warehouses." Proceeding of International Conference on Control, Automation and Diagnosis, IEEE Publications.

[4] Rania Yangui, Ahlem Nabli, Faiez Gargouri, 2016, "Automatic Transformation of Data Warehouse Schema To NoSQL. " Elsevier publication".

[5] Max Chevalier, Mohammed El Malki, Arlind Kopliku, ….., 2016, "Document-oriented data warehouses: Models and extended cuboids, extended", Tenth IEEE International Conference on ResearchChallenges in Information Science, RCIS 2016, IEEE, 1–11.

[6] Kwangchul Shin, Chulhyun Hwang, Hoekyung Jung. "NoSQL Database Design Using UML. " Research India Publications.

[7] Blaha, Michael, 2013, "UML Database Modeling Workbook." Technics Publications.

[8] Christine Niyizamwiyitira and Lars Lundberg, "Performance Evaluation Of SQL and NOSQL DataBase Management Systems in a Cluster", International Journal of Database Management Systems (IJDMS) Vol.9, No.6, June 2017

## AUTHORS

Professor and Researcher in Computer Science and Information System in the Faculty of Science and Technology of Settat, Hassan University 1 Morocco.