A Unified Conceptual Model for Data Warehouses

Shreya Banerjee *
Eastern International University,
Binh Duong Province, Vietnam,
shreya.banerjee@eiu.edu.vn

Sourabh Bhaskar
Parul Institute of Engineering & Technology,
Parul University, Vadodara, India,
sourabhb440@gmail.com

Anirban Sarkar
National Institute of Technology Durgapur,
Durgapur, India,
sarkar.anirban@gmail.com

Narayan C. Debnath
Eastern International University,
Binh Duong Province, Vietnam,
narayan.debnath@eiu.edu.vn
Overview

01 Motivation and Contribution

02 Proposed Conceptual Model for NoSQL Based DWs

03 Illustration of the proposed conceptual model using a case study

04 Implementation Strategy
Introduction

- Modern Data Warehouse (DWs) are able to handle and excel with new trends in data analysis such as
  - rapid growth of data, fast query expectations from users,
  - non-relational/unstructured data, and cloud born data
- Traditional DW and On Line Analytical Processing (OLAP) are comprised of a set of concepts like,
  - facts, dimensions, measures and dimension hierarchies, those are used for structured schema representations
  - cube is used for multi-dimensional data visualization
- In case of web-scale applications, many of the dimensional information may not be available in regular structure
Motivation

- Decision makers are increasingly using NoSQL databases as a deployment tool for Data Warehouses (DW)
  - due to its support for dynamic and scalable data modeling

Capabilities

- NoSQL databases are classified based on different physical level data models –
  - Key-Value stores, Document Store, Column-Family store and Graph databases

- This heterogeneity brings several dimensions of challenges in
  - systematic design methodology for NoSQL based DW solutions
Motivation

Significant challenges among those are

- Lack of common conceptual model for different NoSQL databases
- NoSQL based implementation of DWs requires a systematic design methodology
  - different levels of abstraction in DW design including conceptual level, logical level and physical level
- Representation of agreeable numerical data (facts and measures) and contextual data (dimensions and its hierarchies)
  - to illustrate the effective associations among Fact, Measure and Dimension
- De-normalization of both contextual and numerical data
  - to achieve flexible characteristics of NoSQL databases
- Realizations of data cubes in NoSQL solutions
  - important for visualizing and executing analytical queries
Contribution

- Proposed work is addressing the mentioned challenges
- An ontology driven common conceptual model for NoSQL based DW system is proposed
  - Ontology is defined as an explicit specification of shared conceptualization of the elements of DW domain in terms of concepts and related axioms

Figure 1. Proposed systematic design methodology for NoSQL based data warehouses
Proposed Conceptual Model for NoSQL Based DWs

- Proposed conceptual model is consisting of common set of constructs, relationships and a number of significant properties:
  - unify conceptual level representations of different NoSQL based DW solutions
  - consisting of all details necessary for representation the concepts of facts, dimensions and measures in DW
  - provides the concepts of data cubes and dimension hierarchies
    - when multi-dimensional data are heterogeneous types, and ranged from structured to semi-structured
  - The proposed conceptual model is equally useful for traditional DW modelling
  - All concepts in the proposed model are represented through axioms expressed using mathematical logic
Proposed Conceptual Model for NoSQL Based DWs

Figure 2. Proposed conceptual model for NoSQL based data warehouse
Constructs and Layers in proposed conceptual model

- Proposed conceptual model can be realized as a layered organization composed of three main layers namely-
  - Collection, Family and Attribute
- Three layers have their respective construct types- Collection (col), Family (FA), and Attribute (AT)
- Fact and dimension hierarchies in DW map towards Family layer
- The measure and members of dimensions are mapped towards Attribute layer
- Collection layer realizes the data cubes based on facts
Constructs and Layers in proposed conceptual model

- **Attribute** is the base layer of the proposed conceptual model
- **Attribute** (AT) construct type is the group of all possible instances of a data item
  - AT is elementary in nature
  - \( \forall x (AT(x) \rightarrow (M_{AT}(x) \oplus D_{AT}(x))) \)
- This can be of two types namely- **Measure Attribute** \((M_{AT})\) and **Dimension Attribute** \((D_{AT})\)
- A \(M_{AT}\) represents single measure of a fact in a DW
- A \(D_{AT}\) represents single attribute belonging to a dimension in a DW
Constructs and Layers in proposed conceptual model

- *Family* is the middle layer of conceptual model
- Several semantically related *AT* are grouped together to form an *Family* (FA) construct type
- *Family* can be of two types –
  - *Fact Family* (FF) and *Dimension Family* (DF)
  - \[ \forall x \exists r \exists v (FA(x) \leftrightarrow (AT(v) \land Cnt_{FF}(r) \land r(x, v) \land (FF(x) \lor DF(x)))) \]
- An FF has single level
- FF comprises of related topmost layer DFs and a set of *MAT* defined on measures
  - \[ \forall x \exists r1 \exists y \exists v \exists r2 (FF(x) \leftrightarrow (Cnt_{FF}(r1) \land MAT(v) \land r1(x, v) \land AS(r2) \land DF(y) \land r2(x, y))) \]
Constructs and Layers in proposed conceptual model

- \( DF \) can be decomposed into multiple levels as per the designer’s choice
  - Multiple levels in \( DF \) represent hierarchies in dimensions
  - The lowest level \( DF \) will exhibit the high level of granularity in multidimensional NoSQL databases
  - composed from the set of \( D_{AT} \) only
  - The higher layer \( DF \) in the dimension hierarchy is the combination of one or more \( D_{AT} \) and associated \( DFs \) of adjacent inner layer
  - \( \forall x_1 \exists x_2 \exists r_1 \exists r_2 \exists r_3 \exists v_1 \exists v_2 ((DF(x_1) \land DF(x_2)) \rightarrow (Cnt_{DF}(r_1) \land Cnt_{DF}(r_2) \land Icnt_{DF}(r_3) \land D_{AT}(v_1) \land D_{AT}(v_2) \land r_1(x_1, v_1) \land r_2(x_2, v_2) \land r_3(x_1, x_2) \land notEqual(x_1, x_2))) \)
 Constructs and Layers in proposed conceptual model

- *Collection* is the top most layer of the conceptual model
- Semantically related *FF* are assembling to form a *Collection (col)* type
- From the top level the entire DW can be viewed as set of *Collections*
  
  \[
  \forall x \exists r \exists v (col(x) \leftrightarrow (FA(v) \land Cnt_{col}(r) \land r(x, v)))
  \]

- *Cube* is the de-facto logical representation for data visualization
  
  - *Cube* can be created from *FF*
  - realized as a *col* in the proposed conceptual model
  - If there are multiple *FF*, then a cube can be devised for each *FF* or combinations of *FF*

  \[
  \forall x \exists r1 \exists e \exists v \exists r2 \exists k \exists r3 (cube(x) \leftrightarrow (FF(v) \land DF(e) \land M_{AT}(k) \land Cnt_{col}(r1) \land Cnt_{FF}(r2) \land AS(r3) \land r1(x, v) \land r3(v, e) \land r2(v, k)))
  \]
Relationships in proposed conceptual model

- Distinct constructs of proposed conceptual model are connected with each another using different relationships
  - These relationships are two types
  - One is inter-layer kind relationships and another is intra-layer kind of relationships
  - Inter-layer kind relationships exist between disparate construct types of two different layers
  - Intra-layer kind relationships exit between analogous construct types of identical layer

- **Containment** (*Cnt*) relationships exist when one construct type encapsulates another construct type. *Cnt* can be present between
  - (i) one *col* and several *FF*, (ii) an *FF* and several *MAT* and (iii) a *DF* and several *DAT*
  - *Cnt* relationships can be both *inter-layer kind* and *intra-layer kind* relationship
  - \( \forall r \exists y \exists^n z (Cnt_{col}(r) \leftrightarrow (Col(y) \land FF(z) \land r(y, z) \land (greaterthanequal \ (value(n), 1)))) \)
Relationships in proposed conceptual model

- **Inverse Containment (Icnt)** relationship is *intra-layer* kind
  - connects two construct types when one is encapsulated towards another construct type dynamically
  - Direction of this relationship is opposite to the *Cnt* Relationship
  - lower level *DFs* are encapsulated towards higher-level *DFs* using *Icnt* relationships
  - This relationship is helpful to represent distinct levels of granularity in dimension hierarchies
  - It is capable to add different dimensions in distinct granular level on the fly
  - useful to change granularity level dynamically
  - \[ \forall x \exists y \exists^n z (Icnt_{DF}(r) \leftrightarrow (DF(y) \land DF(z) \land DF_{level}(y) \land DF_{level\_next}(z) \land r(z,y) \land (greaterThanEqual(value(n, 1)))) \]
Relationships in proposed conceptual model

- **Association (AS)** is intra-layer kind relationship and connect constructs types intended for accomplishment of several objectives together

- An AS may exist between
  - FF and DF
  - two different *cols*

- Proposed relationships have different properties
  - **Cardinality**, **Modality** and **Ordering**

- **Cardinality** defines numbers of participate instances
- **Modality** defines optional and/or mandatory participation
- **Ordering** defines whether the constructs participating in a relationship are in order or not
  - if value of *Ord* is 1, then participants are in order
  - if value of *Ord* is 0, then participants are not in order
Relationships in proposed conceptual model

- There can be different values for $Crd$ and $Mdl$. Those are:
  - 1:1 – Represents $AT$ and $FA$ relationship with mandatory total participation.
  - 0:1 – Represents $AT$ and $FA$ relationship with optional one participation.
  - 1:M – Represents $AT$ and $FA$ relationship with mandatory multiple participation.
  - 0:M – Represents $AT$ and $FA$ with optional multiple participation in the relationship.
  - 0:X – Represents $AT$ and $FA$ with optional exclusive participation in the relationship.
  - 1:X – Represents $AT$ and $FA$ relationship with mandatory exclusive participation.

Contd.
Illustration of Proposed OLAP Algebra Using a Case Study

- The case study is based on sales and shipping
  - Sales of different products can be done in sale branches
  - Branches can be located in multiple locations
  - Shipping can have multiple shippers who will shipped the product from one location to another
- This case study has two facts – Sales and Shipping
  - Sales is associated with four dimensions - Location, Branch, Product, Time
  - Shipping is associated with four dimensions - Location, Shipper, Product, Time
  - Two facts share three dimensions - Location, Product, and Time
  - Several dimensions have hierarchy : Time→Day→Month→Year
several dimensions have specific attributes

- *Time Id*, and *Time*

In addition, each fact are associated with two measures

- Shipping is associated with measures Units Shipped and Dollars Cost

In some cases, attributes of specific dimension is either changed or absent

- Distinct features of this described data set is highly irregular and require flexible representation.
- NoSQL databases are required to demonstrate these data set in DWs
Illustration of Proposed OLAP Algebra Using a Case Study

Collections (Cubes created from Fact Families)
FACT FAMILY 1 (SALES)
FACT FAMILY 2 (SHIPPING)
SALES(Location, Branch, Product, Time, units sold, dollars sold)
SHIPPING (Location, Shipper, Product, Time, units shipped, dollars shipped)
Location (location_Id, pin code, {street}, city_Id)
City (city_id, city, state_Id)
State (state_Id, state, country_Id)

Country (country_Id, country)
Branch (branch_Id, branchName)
Product (product_Id, product_Name, productType_Id)
ProductType (productType_Id, productType_Name)
Time (time_Id, time, day_Id)
Day (day_Id, day, month_Id)
Month (month_Id, month, year_Id)
Year (year_Id, year)
Shipper (shipper_Id, shipperName, location_Id)

Nomenclature
Collections: In Capitalize and **bold**;
Fact Families: in UPPERCASE and *italic*
Dimension Families: in Capitalize and italic
Measure Attributes: in lowercase and italic
Dimension Attributes: in lowercase
Optional Construct Type: within {}
Illustration of Proposed OLAP Algebra Using a Case Study

Figure 4. Shipping and Sales Fact Family with related Dimension Families and Measure Attributes
Implementation Strategy

- Two kinds of strategies are proposed for implementation of data cubes in NoSQL based DW systems
  - Single Collection based Implementation Strategy
  - Multiple Collection based Implementation Strategy

- In Single Collection based Implementation Strategy,
  - data cubes will be realized as a single col of a FF
  - numbers of data cubes in DW system depend on numbers of FFs

- In Multiple Collection based Implementation Strategy,
  - A data cube can be realized based on multiple cols of FFs and related DFs
  - These multiple cols include cols of each DFs related with a FF and a col of the FF itself
  - data cubes will be devised dynamically (on the fly)
  - This strategy is capable of creation of flexible schema for NoSQL based DWs
### Table 1. Comparison Table between Multiple Collection based and Single Collection based Implementation

<table>
<thead>
<tr>
<th>Multiple Collection based Implementation</th>
<th>Single Collection based Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>This strategy has less redundancy, because, a fact or shared dimensions are defined once.</td>
<td>This strategy has high-level redundancy, since a fact or shared dimensions are defined multiple times.</td>
</tr>
<tr>
<td>After defining once, insertion of new data definitions are propagated to other places. Hence, addition of data definitions can be handled easily.</td>
<td>Addition of data definitions is costlier.</td>
</tr>
<tr>
<td>Maintenance is inexpensive than single one.</td>
<td>Maintenance is expensive than multiple one.</td>
</tr>
<tr>
<td>Due to more data integration policy, query execution time will be higher.</td>
<td>Due to less data integration policy, query execution time will be lower.</td>
</tr>
</tbody>
</table>
# Mapping towards MongoDB

<table>
<thead>
<tr>
<th>Constructs of proposed conceptual model</th>
<th>Equivalent MongoDB representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection construct type</td>
<td>Collection</td>
</tr>
<tr>
<td>Fact family construct type</td>
<td>Document</td>
</tr>
<tr>
<td>Dimension family construct type</td>
<td>Document</td>
</tr>
<tr>
<td>Dimension attributes</td>
<td>Field</td>
</tr>
<tr>
<td>Measure attributes</td>
<td>Field</td>
</tr>
<tr>
<td>Association</td>
<td>Represented using Nested document</td>
</tr>
<tr>
<td>Containment</td>
<td>Represented using Nested document</td>
</tr>
<tr>
<td>Inverse Containment</td>
<td>Dynamic insertion of document towards another document without specifying its schema</td>
</tr>
<tr>
<td>Cardinality</td>
<td>1: M:-The construct type whose participation is M will be the nested document or nested member field of a parent document or field, whose participation is 1. 1: 1:- Any one of two construct types can become nested document or nested member field of another construct type.</td>
</tr>
<tr>
<td>Optional modality</td>
<td>Flexible modality of all relationships.</td>
</tr>
<tr>
<td>Ordering</td>
<td>Ordered set is mapped towards “Array” and unordered set mapped towards “document”.</td>
</tr>
</tbody>
</table>
Mapping towards MongoDB

Collection of Shipper Dimension (collection1)
{ "_id" : ObjectId("5a140bd51b94b042474b8fd7"), "Shipper" : { "shipper_Id" : 501.0, ------- }

Collection of Location Dimension (collection2)
{ "_id" : ObjectId("5a155ec53b820e506813c9f9"), "Location" : { "location_Id" : 101.0, "pincode" : 713209.0, ------- }

Collection of Product Dimension (collection3)
{ "_id" : ObjectId("5a1560733b820e506813c9fc"), "Product" : { "product_Id" : 301.0, "product_Name" : "TV", ------- }

Collection of Time Dimension (collection4)
{ "_id" : ObjectId("5a1561643b820e506813c9fe"), "Time" : { "time_Id" : 401.0, "date" : ISODate("2017-11-"}, "Time" : { "time_Id" : 401.0, "date" : ISODate("2017-11-"}, "Time" : { "time_Id" : 401.0, "date" : ISODate("2017-11-"}, "Time" : { "time_Id" : 401.0, "date" : ISODate("2017-11-"}, "Time" : { "time_Id" : 401.0, "date" : ISODate("2017-11-"}, "Time" : { "time_Id" : 401.0, "date" : ISODate("2017-11-") }

Collection of Shipping Fact
{ "_id" : ObjectId("5a17b7728d4ca0e7112c7931"), "location_Id" : 101.0, "shipper_Id" : 501.0, "product_Id" : 301.0-- }

Figure 7: MongoDB based implementation for Collections of Shipping fact and related dimensions
Mapping towards MongoDB

Figure 5. Single Collection based Implementation Strategy in MongoDB based on the specified case study

db.shipping.aggregate([{"$lookup":{"from":"shipper","localField":"shipper_Id","foreignField":"Shipper.shipper_Id","as":"collection1_doc"}},
{"$unwind":"$collection1_doc"}],

OUTPUT:

Figure 6: Multiple Collection based Implementation Strategy in MongoDB for the specified case study
### Conclusion and Future Work

- **Novelties of the proposed work**
  - A systematic methodology for implementation of NoSQL based DWs
  - A generalized and rigorous formal conceptual model
  - Realization of flexible characteristics of NoSQL based DWs by de-normalizing both contextual and numerical data
  - Handling dimension hierarchies at different granular levels
  - Multiple implementation strategies and visualization techniques of data cubes over NoSQL based databases
  - Realization of traditional DWs
    - When ordering and modality of distinct relationships are strictly set to 1 and Inverse Containment relationships do not exist

- **Future Work will include**
  - Performance evaluation and validation
  - Automated transformation mechanism for proposed conceptual model into specific physical databases
THANK YOU