Extended Change Identification System

Parimala N.[1] and Vinay Gautam[2]
[1]School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi 110067, India
[2]Chandigarh Engineering College, Landran, Mohali, Punjab 140307, India

Keywords: Multi-version Metadata, Metadata, Ontology, Data Warehouse.

Abstract: Schema evolution leads to multiple versions of the data warehouse schema. We address the issue of whether the information required by the decision maker is present in some version of the data warehouse or not by checking all the versions for the existence or the absence of the required information. The user specifies the sought information using business terms. We build Delta Ontology which captures the ontology information in terms of mapping between business terms and schema terms. The Delta Ontology is built for the modifications to the schema as it evolves. We propose an algorithm to search for the information in the latest version of E-Metadata and the Delta Ontology. Our algorithm lists all the versions where the information is available giving precedence to finding the information in a single version over across versions. The decision maker is also informed if the information is totally missing.

1 INTRODUCTION

We are looking at the definition of the warehouse per se and analyzing its adequacies to meet the needs of the user. If the current data warehouse does not meet the needs of the user then it implies that there is a gap between the information content of the warehouse and the information that is needed by the decision maker. Further, the latest warehouse schema could have evolved over time. That is, there could be changes made to the schema more than once. It is possible that the information sought by the user, which is absent in the current schema, existed in some earlier version of the schema and has been subsequently deleted. If it is possible to trace the changes, then this trace can help the user in analyzing the manner in which the gap has arisen.

Consider an example of an Insurance Schema represented as a star schema shown in Figure 1. The Multi-dimensional schema has Policy_holder, Policy, Claim and Time as dimensions. Policy Revenue is the fact with Premium_dollar and Coverage_period as measures. Let the schema in Figure 1 be the latest version, version3. Let us assume that an attribute Holder_gender was present in version 1 and was subsequently deleted. It is, therefore, not present in version2 and version3.

Consider the case where it is needed to get "the revenue that was generated gender wise for each city". If we look at the latest version of the schema as given in Figure 1,we see that 'revenue' is defined as premium_dollar and there is no dimensional attribute 'gender' in version3. However, the attribute, 'gender', was defined as Holder_gender in the earlier versions. Further, the attribute 'city' is part of the attribute Holder_address. In other words, three things can be observed.

1. The decision maker is expressing the information using business terms which are not the same as schema terms.
2. The information may not be directly represented. In the above example, the attribute city is not directly represented but is part of the attribute address. Different cases arise when information is not directly represented. These have to be identified.
3. The sought information could be missing. Information could be missing either because it is not defined in the latest version and existed in an earlier version since the schema has evolved over time or because it is altogether absent.

We briefly outline the solutions to the problems listed above.

1. Ontology is defined consisting of business as well as schema terms and a mapping between these. The schema terms of all the schema versions are grouped into two groups – the terms

N., P. and Gautam, V.
Extended Change Identification System.
In *Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering (ENASE 2016)*, pages 51-58
ISBN: 978-989-758-189-2

that have remained constant across versions and those that have participated in the evolution. Delta Ontology (DO) has ontology entries for the changed schema terms across versions.

2. A set of rules are defined to identify the form in which the information is present in the latest version or existed in the earlier versions.
3. The third issue is addressed by checking whether the information existed in any earlier definition of the schema.

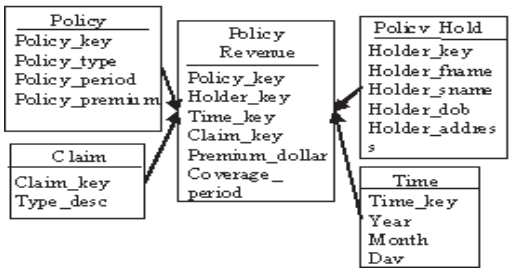

Figure 1: Insurance Warehouse Schema.

Our system, known as the eXtended Change Identification System (X-CIS), is shown in Figure 2. First, we build the DO. Next, the Information Processing component searches for the terms specified by the user across schemata. The ontological entries for the constant terms are available in the latest version of E-Metadata (section 2.2). Thus, instead of a blind search across versions, we search in the latest E-Metadata and DO.

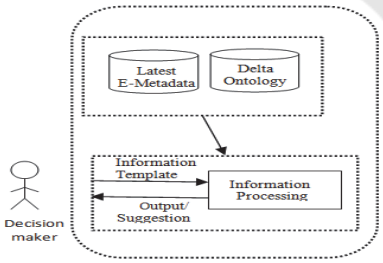

Figure 2: X-CIS Architecture.

The contribution of the paper is three fold: Firstly, we show the manner in which Delta Ontology is built. Secondly, we propose Delta Scan Algorithm (DSA) to search for the requested information across data warehouse versions. Thirdly, we identify the forms in which the requested terms are present.

1.1 Related Work

Conventional approaches for the management of changes to a multidimensional schema and the contents (which is the data warehouse) can be broadly classified into two categories namely, schema evolution where the changes are made to the multidimensional structure without retaining the existing definition (Blaschka et al., 1999; Kaas et al., 2004; Bebel et al., 2006) and version extension, where all the versions are maintained (Body et al., 2002; Shazad et al., 2005; Golfarelli, 2006).The evolution in a data warehouse schema affects the data warehouse metadata as described in (Vaduva and Dittrich, 2001; Pan, 2010).

Different versions of data warehouse metadata have been maintained and queried for different purposes. In (Wrembel and Bębel, 2007) the user is informed about the missing attributes in a query by means of meta-information that is attached to the result. The user is also informed about the changes in the structure. In (Leja et al., 2010) the query language, MVDWQL, supports two types of queries, namely content queries and metadata queries through a GUI. Multiple version data can be queried using the former. The history of evolution can be sought by executing a metadata query. A set of traces which relate multidimensional elements and data sources is maintained in (Maté and Trujillo, 2014). With the help of these traces, changes to the schema when data sources change are easily incorporated. In all these systems, the user queries are restricted to using schema terms.

In this paper, we explore the schema definitions in the different versions and the history of changes to find the versions where the information sought by the user is present. The user's need is expressed using business terms. Towards this, we build delta ontology using the changes reflected in the multi version metadata. Our work is different in that the information sought by the user is expressed in business terms and not warehouse schema names. Secondly, we do not query multiple metadata versions but build Delta Ontology which records only the changes.

The rest of the paper is organized as follows. Definitions are given in section 2.In section 3, the creation of Delta Ontology is explained. The X-CIS architecture is explained in section 4. Section 5 is the concluding section.

2 DEFINITIONS

In this section, we define the terms used in this paper.

2.1 Analysis Component

The information needed by the decision maker is as given in (Parimala and Gautam, 2010). It is expressed using two terms 'what' and 'how' where
a)'what' or "what is to be analyzed" describes the data to be analyzed. It refers to the measures which represents the factual data.
b)'how' describes the business perspective under which data analysis is to be performed. It specifies the dimensional attributes along which the measure is to be analyzed.

2.2 E-Metadata

E-Metadata consists of the technical metadata of the data warehouse schema and the ontology as defined in (Parimala and Gautam, 2011). The technical metadata is extracted from the metadata of a warehouse schema. The ontology for the terms in the technical metadata is built using the WordNet. Figure 3 shows the partial E-Metadata entry for Policy_Holder. The entry says, for example, that customer is a synonym of Policy_Holder.

```
<Dimension-3-Policy_Holder>
<Synset-of-Dimension-3>Syn-
customer</Synset-of-Dimension-3>
<Term-of-Dimension-3>Meronyms-</Term-
of-Dimension-3>
<Hierarchy-of-Dimension-3>Hiper-Enter
Hypernym of Policy_Holder</Hierarchy-
of-Dimension-3>
</Dimension-3-Policy_Holder>
```

Figure 3: E-Metadata Entry.

3 DELTA ONTOLOGY

Delta Ontology (DO) is expressed using OWL. DO has six ontology classes which are DVersion, Operation_type, DWTerm, Label, Synset and Term. DVersion contains the version of a delta file. Corresponding to each term of an entry in the delta file (as shown above) we define three classes namely Operation_type, DWTerm and Label. In addition, two new classes Synset and Term are introduced where Synset class denotes the domain terms which have the same sense as the terms in Label class and Merset class denotes meronyms of terms in the Label class.

The relation between the classes in the ontology is represented as OWL Property. Relations in DO show the associations between the classes. These relations are Oper_DWT, Oper_Label, Label_DWT, Label_Synset ,Label_Merset and DVersion_Oper. Oper_DWT is a relation between the classes Operation_type and DWTerm which shows whether a DWTerm is added or deleted. The rest of the relations are self explanatory.

3.1 Delta Ontology Development

We create a repository of the changes that have taken place. As brought out in (Pan, 2010; Saddad et al., 2008), the changes that the user can make to the technical metadata schema are insertion, deletion and renaming of any of a dimension, a dimensional attribute and a fact attribute.

A new version of the technical metadata reflects the changes to the data warehouse schema. The Delta file, $D_{i,i+1}$, shows the difference between technical metadata version i, TM_i and technical metadata version i+1,TM_{i+1} (Gautam and Parimala, 2012). The Delta Ontology is built by extracting the information from the delta file. The changes can be addition, deletion or renaming of schema names. In the new version of the technical metadata, the newly added elements are inserted; the deleted ones are not reflected. As far as rename is concerned the new name is added to the new version and the old name exists in the previous version. Thus, an entry in the delta file is of the form:

<Operation_type DWTerm = Label [DWTerm = Label2]>

Here, Operation_type is one of insertion, deletion or rename. It reflects the operation used to perform the movement from one version to the next. DWTerm is either a Fact, Dimension, Measure or Dimension Attribute. Label is a term from a warehouse schema such as city, policy etc. As an example of an entry consider

Example 1: <Insert Attribute='Quarter'>

which says that the attribute 'Quarter' has been inserted.

The optional specification, 'DWTerm = Label2', is valid only when Operation_type is *Rename*. For e.g. the following entry in the delta file shows that 'Holder_sname' is renamed as 'Holder_lastname'.

Example 2: <Rename Attribute = "Holder_sname" Attribute="Holder_lastname">

The Delta Ontology Development Process (DODP) starts by creating an instance of DVersion. The instance contains the version of the delta file

$D_{i,i+1}$. Next, Java API is used to extract the terms, referred to as a 'token', from the delta file $D_{i,i+1}$.

In the next step, the base class for each 'token', from among the DO classes (section 4), is identified. The following rules are used to identify the base class for a given token.

```
R1: If (token =  Operation_type) then
   "Base Concept of token is
   Operation_type"
R2: If (token =  DWTerm) then "Base
   Concept of token is the DWTerm"
R3: If (token =  Label) then "Base
   Concept of token is the Label".
```

The token itself, is added as an instance of the base class to which it belongs. If the 'Operation_type' is 'Rename, then it is treated as if the old Label is deleted and the new Label has been inserted. Therefore, the system will create two instances of Operation_type which are 'Delete' and 'Insert'.

Consider Example 1. The extraction process will extract three tokens namely Insert, Attribute and Quarter. The token 'Insert' is added as an instance of Operation_type', 'Attribute' as an instance of 'DWTerm' class and 'Quarter' as an instance of 'Label' class. In Example 2, both 'Holder_sname'and 'Holder_lastname' become instances of Label class. Since the operation is rename, two instances of Operation_type', 'Insert' and 'Delete' are created.

Subsequently, the domain concepts are added using WordNet. In this step, the WordNet is used to extract terms (Synnonym and Meronyms) for the tokens belonging to the class 'label.' Synonyms are words with similar meaning and meronyms show partof relationship of a token with WordNet terms. The synonyms are added as instances of Synset; the meronyms are added as instances of Term in the DO.

The above information can be added provided the token is found in the WordNet (Canas et al., 2003; Miller and Hristea, 2006; Miller, 1995). If the token is not found in the WordNet, then the Data Warehouse Administrator (DWA) is asked to specify an equivalent term which can be found in the WordNet. The approach of asking for an equivalent term was prompted by studying different example schemas. (http://merc.tv/img/Figure./ Adventure WorksDW2008.pdf, http://www.information-management-architect.com /star-schema .html). The schema term CalenderYear has no entry in WordNet. However, the word Year is present in WordNet. If the DWA can specify, Year as an equivalent term then it enhances the ontology.

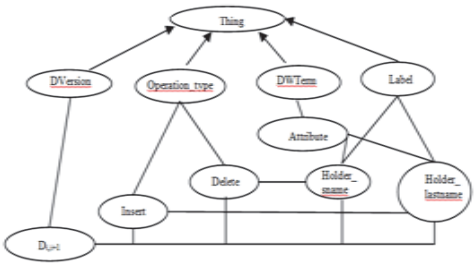

Figure 4: Delta Ontology after Updates.

Next, the relations between the instances are established. All the instances created above are linked to their respective classes. All the instances are also linked to the DVersion instance. Further, all the instances of the Label class are linked to the corresponding instance of Operation_type. Figure 4 shows DO for Example 2.

4 X-CIS ARCHITECTURE

Once the Delta Ontology is built, eXtended Change Identification System (X-CIS) identifies the form and the versions where the requested terms are present. The X-CIS architecture is shown in Figure 2. The input to the system is 'what' and 'how' terms specified as the analysis component. X-CIS uses the latest version of E-Metadata and DO to search for the information. The output of the system is either the versions where the data is available or a suggestion for a change in the warehouse schema, if the term is absent.

The manner in which E-metadata is searched is as given in (Parimala and Gautam, 2010). Here, we explain the Delta Scan Algorithm (DSA) which looks for the terms in the DO. Before explaining DSA, the guidelines for identifying the forms in which the terms may appear is given.

4.1 Forms of Terms

The existence of an analysis component term in the DO has two aspects to it. The first is whether the term is directly or indirectly available. The second is the version where it is available since the search spans across versions. We consider each of these in turn before explaining the order in which the result is presented to the user.

Directly Available

If the 'what' term is defined as a measure in the E-Metadata, we say that it is directly available. The corresponding statement is

WS1: is a measure

Similarly, the 'how' term is directly defined if it is the name of a dimensional attribute. The corresponding statement is

HS1: is a name of an attribute of a dimension.

Indirectly Available

If a term is not directly available, then, it may be defined in the schema but not as expected or it may be an ontological equivalent. In these cases, we say that the term is indirectly available. The different ways in which 'what' can be indirectly available is as follows:

WS2: is a synonym of a measure
WS3: is the name of the fact table.
WS4: is a dimension name.
WS5: is a name of an attribute of a dimension.

The different ways in which 'how' is indirectly available is as follows:

HS2: is a synonym of a dimension attribute.
HS3: is a meronym of a dimension attribute.
HS4: is a dimension name.
HS5: is a fact.
HS6: is a fact attribute (measure).

Table 1: Result Options

Sl. No.	'what' term	'how' term
1	D_i	D_i
2	D_i	D_j
3	I_i	D_i or I_i or I_j
4	D_i or I_i or I_j	I_i
5	D_j or I_i	N
6	N	D_j or I_i

Version Possibilities

Consider, now, the second aspect of finding the 'what' and 'how' terms across versions. Both the terms may be defined in the same version, either directly or indirectly or they me be defined in different versions, again directly or indirectly. Table 1 shows these different possibilities. If the 'what' or the 'how' term is directly available in version i then it is denoted as D_i; ID_i implies that the term is indirectly available in version i. N says that the term is not available.

Message

The result of searching a 'what' term is expressed as WS_kV_i = 'what term is available as WS_k in version i' and that of 'how' term is HS_kV_i = ' how term is available as HS_kin version i'.

Result Order

When a term is available in more than one version, instead of just listing the versions, we order the result based on our belief that users would prefer to find 'what' and 'how' in the same version to finding 'what' and 'how' across versions. This is because if the information is found in the same version then the user can query the corresponding data warehouse. X-CIS will display the information according to the preference order given below.

Preference 1: Both are directly available in the same version (Sl. No. 1 of Table 1).

Preference 2: Both are directly available across versions (Sl. No. 2 of Table 1).

Preference 3: Same or across versions, one of them is directly available (Sl. No. 3 and 4 of Table 1).

Preference 4: One of them is available either directly or indirectly and the other is not available (Sl. No. 5 and 6 of Table 1.)

If the information is not available in any version (Sl. No. 7 of Table 1), then we suggest that changes may have to be made to the current version.

4.2 Delta Scan Algorithm

Let V_1 ,V_2,…,V_n be the E-Metadata versions with $V_n$as the latest version. DSA has two parts. The first is the creation of the two matrices WAvail and HAvail. The second is to scan the matrices and inform the user about the evolution.

In the rules given below, IsEqual( ) is a Boolean function which compares two terms and returns true if they refer to the same term; otherwise it returns false (Parimala and Gautam, 2010).

The rules given below set the values in the matrices WAvail and HAvail. The entries in Delta Ontology for a 'what' or a 'how' term specify the operation(Insert or Delete) which created the term which is an instance of the Label class. The operation is linked to an instance of DVersion. If the operation is Insert, then it is available in the higher version; if it is Delete, then the term is available in the lower version. The version instance is of the form $D_{i,i+1}$. Thus, if the operation is Insert, then it implies that the term is now available in the schema version i+1. On the other hand, if the operation is Delete, then it means that the term was available in schema version i. In the former case, (i+1)th row of the matrices is set to 1 and ith row in the latter case. Consider, next, the columns. A column in WAvail is set depending on the considerations of a match between 'what' term and other schema terms as given in section. Similar considerations for the 'how' term determine which column of HAvail is set to 1. If any rule fires, then the remaining rules are skipped.

4.2.1 Matrix Creation

Rules for 'what'

```
1) If the 'what' term is an instance of
   'Label' class and related version
   DVersion = i,i+1 then
    If
IsEqual('what','Fact_Attribute')then
      If Operation_type ='Insert' then
setWAvail[i+1,1] to 1
      If Operation_type ='Delete' then
setWAvail[i,1] to 1
    If IsEqual('what','Fact') then
      If Operation_type ='Insert' then
setWAvail[i+1,3] to 1
      If Operation_type ='Delete' then
setWAvail[i,3] to 1
    If IsEqual('what','Dimension') then
setWAvail[i+1,4] to 1
      If Operation_type ='Insert' then
setWAvail[i+1,4] to 1
      If Operation_type ='Delete' then
setWAvail[i,4] to 1
    If  IsEqual('what','Dim_Attribute')
then
      If Operation_type ='Insert' then
setWAvail[i+1,5] to 1
      If Operation_type ='Delete' then
setWAvail[i,5] to 1

2) If the 'what' term is an instance of
   'Synset' class then the related
   instance of 'Label' is picked up. If
   related version DVersion = i,i+1
   then
     If     IsEqual('related     Label
instance', 'Fact_Attribute') then
      If Operation_type ='Insert' then
setWAvail[i+1,2] to 1
       If Operation_type ='Delete' then
     setWAvail[i,2] to 1
```

Rules for 'how'

```
1) If the 'how' term is an instance of
    'Label' class and related version
    DVersion = i,i+1 then
    If  IsEqual('how',  'Dim_Attribute)
then
      If Operation_type ='Insert' then
setHAvail[i+1,1] to 1
      If Operation_type ='Delete' then
setWAvail[i,1] to 1
    If IsEqual ('how','Dimension') then
      If Operation_type ='Insert' then
setWAvail[i+1,4] to 1
      If Operation_type ='Delete' then
setWAvail[i,4] to 1
    If IsEqual ('how','Fact') then
      If Operation_type ='Insert' then
setWAvail[i+1,5] to 1
      If Operation_type ='Delete' then
setWAvail[i,5] to 1
    If IsEqual ('how', 'Fact_Attribute'
then
      If Operation_type ='Insert' then
setWAvail[i+1,6] to 1
      If Operation_type ='Delete' then
setWAvail[i,6] to 1

2) If the 'how' term is an instance of
   'Synset' class then the related
   instance of 'Label' is picked up. If
   related version DVersion = i,i+1
   then
    If    IsEqual    ('related    Label
instance', 'Fact_Attribute') then
      If Operation_type ='Insert' then
setWAvail[i+1,2] to 1
      If Operation_type ='Delete' then
setWAvail[i,2] to 1
```

As an example, let us say that the user wants to analyze 'Revenue' ('what') along 'gender' ('how'). As given in section 1, 'Revenue' is a synonym of the measure Premiun_dollars which is available in all the three versions of the schema. 'gender' as 'Holder_gender' was available only in version 1. The resulting matrices are shown below.

Table 2: WAvail Matrix.

Message / Version	WS1	WS2	WS3	WS4	WS5
1	0	0	1	0	0
2	0	0	1	0	0
3	0	0	1	0	0

Table 3: HAvail Matrix.

Message / Version	HS1	HS2	HS3	HS4	HS5	HS6
1	0	1	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0

4.2.2 Response

Using the matrices built in the previous section, it is possible to inform the user about 'what' and 'how' terms. Recall that in section 3, we showed that the result is of the form WS_kD_ior HS_kD_i. The mapping between the result and the matrices is given below:

The matrices are repeatedly scanned to list the result in terms of preferences.

Preference1 :If WAvail[i,1]=1 and HAwail[i,1]=1 then message is WS_1V_i and HS_1V_i, 1≤i≤ n

Preference2 :If WAvail[i,1]=1 and HAwail[j,1]=1 then message is WS_1V_i and HS_1V_j, 1≤i≤n, 1≤j≤ n, i≠ j

Preference3 :If WAvail[i,1]=1 and HAwail[i,k]=1 then message is WS_1V_i and HS_kV_i,1≤i≤n, 2≤k≤6

If WAvail[i,k]=1 and HAwail[i,1]=1 then message is WS_kV_i and HS_1V_i,1≤i≤n, 2≤k≤5

Preference4 :If matrices WAvailand HAvail are all zeroes then the message is 'what' term and 'how' 'are not available'

If matrix WAvail is all zeroes and if HAvail[i,k]=1 for 1≤k≤5 then the message is 'what' term 'is not available' and HS_kV_i, 1≤i≤n

If matrix HAvail is all zeroes and and if WAvail[i,k]=1 for 1≤k≤6 then the message is 'how' term 'is not available' andWS_kV_i, 1≤i≤n

4.2.3 Time Complexity of DSA

DSA algorithm scans and compares the attributes of the latest E-Metadata version and the DO entries once. The number of comparisons is the number of attributes in the latest E-Metadata (say m) + no. of entries in the Delta Ontology (say k). Therefore,

$$\text{complexity} = O(m+k)$$

In order to analyze the complexity, we compare it with the complexity of an algorithm in the absence of DO. In this case, an algorithm to find 'what' and 'how' terms, would have to scan and compare each attribute of each and every version of E-Metadata. Therefore,

Complexity = average no. of attributes in a E-Metadata version (m) * number of E-Metadata versions (n)

$$= O(m*n)$$

Let us analyze n and k.

Case1: If k is very large and is equal (n-1)*m then the complexity of DSA is the same as complete scan algorithm.

Case 2: If k << (n-1)*m then the complexity DSA is much lower.

Case 1 will occur if the number of changes is as high as the E-Metadata itself. This is a highly unlikely scenario. Case 2 will occur most of the time.

It must be noted that in both the cases matrices have to be built to give the complete picture to the user and further, the time taken to scan the matrices would be the same for both the algorithms.

5 CONCLUSION

The aim is to find whether the multiple warehouse versions cater to the needs of the business user. The needs are expressed using 'what' and 'how' terms. These may be business terms and not necessarily schema terms. We have built Delta Ontology to capture the mapping between business terms and schema terms. The Delta Ontology itself is built using the differences in the metadata of the warehouse schema as it undergoes changes. All the versions where the terms are available is picked up and listed according to the preference order.

Java is used for building the prototype of the system and SQL Server 2005 is used as back end tool to store metadata of the warehouse.

It may be argued that special data structures for sparse matrices can be used to store the contents of WAvail and HAvail. Since the matrices are not very large the time taken to scan them is not very high. As far as populating them is concerned, DSA is better in terms of time complexity than complete scan of all versions.

It may be noted that we have not defined a query language to query multi versions of the metadata. Query language is appropriate when different ad hoc queries are to be framed. In our case, we search only for 'what' and 'how' terms. Therefore, an appropriate GUI is built to accept these terms.

In our system, it is possible for the user to know whether the missing information in the current schema was available in an earlier version or not. This, we believe, will help the user decide whether or not to change the current data warehouse schema. It is, also, possible to see whether earlier versions were more in tune to the decision maker's needs. It will also give a feedback on the evolution of the schema vis-a-vis it satisfying query needs.

REFERENCES

Bebel, B., Królikowski, Z. and Wrembel, R., 2006. Managing evolution of data warehouses by means of nested transactions. In *Advances in Information Systems* (pp. 119-128). Springer Berlin Heidelberg.

Blaschka, M., Sapia, C. and Höfling, G., 1999. On schema evolution in multidimensional databases. In *Data Warehousing and Knowledge Discovery*(pp. 153-164). Springer Berlin Heidelberg.

Body, M., Miquel, M., Bédard, Y. and Tchounikine, A., 2003, March. Handling evolutions in multidimensional structures. In *Data Engineering, 2003. Proceedings. 19th International Conference on* (pp. 581-591). IEEE.

Cañas, A.J., Valerio, A., Lalinde-Pulido, J., Carvalho, M. and Arguedas, M., 2003, October. Using WordNet for word sense disambiguation to support concept map construction. In *String Processing and Information Retrieval*(pp. 350-359). Springer Berlin Heidelberg.

Gautam, V. and Parimala, N., 2012. E-Metadata versioning system for data warehouse schema. *International Journal of Metadata, Semantics and Ontologies*, *7*(2), pp.101-113.

Golfarelli, M., Lechtenbörger, J., Rizzi, S. and Vossen, G., 2006. Schema versioning in data warehouses: Enabling cross-version querying via schema augmentation. *Data & Knowledge Engineering*, *59*(2), pp.435-459.

Kaas, C., Pedersen, T.B. and Rasmussen, B., 2004. Schema evolution for stars and snowflakes. In *Proceedings of the Sixth International Conference on Enterprise Information Systems*.

Leja, W., Wrembel, R. and Ziembicki, R., 2009. On Querying Data and Metadata in Multiversion Data Warehouse. *Data Warehousing Design and Advanced Engineering Applications: Methods for Complex Construction: Methods for Complex Construction*, p.206.

Maté, A. and Trujillo, J., 2014. Tracing conceptual models' evolution in data warehouses by using the model driven architecture. *Computer Standards & Interfaces*, *36*(5), pp.831-843.

Miller, G.A., 1995. WordNet: a lexical database for English. *Communications of the ACM*, *38*(11), pp.39-41.

Miller, G.A. and Hristea, F., 2006. WordNet nouns: Classes and instances. *Computational linguistics*, *32*(1), pp.1-3.

Pan, D., 2010. Metadata version management for DW 2.0 environment'. *Journal of Convergence Information Technology*, *5*(3).

Parimala, N. and Gautam, V., 2010. CIS: Change Identification System. In *KEOD* (pp. 347-350).

Parimala, N. and Gautam, V., 2011. Extended Metadata for Data Warehouse Schema. In *ENASE* (pp. 254-259).

Saddad, E., El-Bastawissy, A., Rafea, M. and Hegazy, O., 2008, March. Multiversion queries in multidimensional structures. In *Proceedings of the 6th International Conference on Informatics and Systems (INFOS'08)*.

Serra, I. and Girardi, R., 2011. A process for extracting non-taxonomic relationships of ontologies from text. *Intelligent Information Management*, *3*, 119.

Shahzad, M.K., Nasir, J.A. and Pasha, M.A., 2005. CEV-DW: Creation and Evolution of Versions in Data Warehouse. *Asian Journal of Information Technology*, *4*(10), pp.910-917.

Vaduva, A. and Dittrich, K.R., 2001. Metadata management for data warehousing: between vision and reality. In *Database Engineering and Applications, 2001 International Symposium on.* (pp. 129-135). IEEE.

Wrembel, R. and Bębel, B., 2007. Metadata management in a multiversion data warehouse. In *Journal on data semantics VIII* (pp. 118-157). Springer Berlin Heidelberg.