Terminological Reasoning and Conceptual Modeling for Datawarehouse

David Rudloff[1]

Abstract. This paper analyses some useful enhancements required by datawarehouse systems. This concerns the conceptual modeling representation and visualisation, the query definition and optimization. A general framework is proposed to integrate terminological reasoning and a natural language interface in a flexible knowledge representation system in order to meet datawarehouse needs.

INTRODUCTION

Nowadays, database access is not exclusively reserved to computer scientists. Many people want to consult directly databases and exploit the data. For these reasons, datawarehouse concerns systems that try to give to non-specialist end-users a simplified view of complex database. Such systems involve database abstraction, query formulation and result analysis. This notion is partially covered by some other business terms, like Enterprise Information System, Business Intelligence, Corporate Analysis System, Online Analytical Processing etc... Let us consider that Datawarehouse involves all the efforts to give a transparent and natural database access for non-specialist end-user. Especially, the aim is to help end-user to search relevant information, to analyze it and edit it, without having to learn some specific database syntax. Concerning relational database, usual access is provided with SQL. It is based on a strong and quite simple theory, the relational model. However, it implies a good knowledge of the relational table structure. Furthermore a simple question can imply a complex query with many joins because of data scattering across the database. In addition, the user may not detect a wrong query with a result that seems right. Thus, user query should be constructable in a natural form, even in natural language. With datawarehouse, we need to construct an conceptual layer upon the physical database structure. This abstract database representation can be customized for each user with a restricted view and appropriate terminology. Another datawarehouse front-end concerns multidimensional information. In this way, hypercubes give the user the ability to aggregate numeric data among various dimensions. Hypercube are usually constructed upon an existing relational database. Then, it could be interesting to explore theoretical aspects of such an abstract layer and the services required to improve current datawarehouse developments. Actually, we would like to integrate the conceptual modeling usually made during the database design in a datawarehouse system. This presents some advantages for consistency check, query optimization, database access improvement and natural language.
The first section of this paper will show some specific lacks in current datawarehouse systems from the end-user point of view.
Then, we will propose, in the second section, a general framework that could reach these requirements.

[1] ERIC-LIIA (ENSAIS), and Neurone Informatique, Strasbourg, F
24, boulevard de la Victore 67000 Strasbourg
Tel: (33) 88.14.47.43. Email: rudloff@.eric.u-strasbg.fr

USEFUL REQUIREMENTS FOR DATAWAREHOUSE

One important datawarehouse requirement concerns the conceptual model. We will first present some ambiguities concerning the distinction between concept and relation implied by naming convention. We will then see that a powerful conceptual representation should show to the end-user the connections between database entities in a natural and flexible form. Next, the subject of query construction and optimization will be tackled in the natural language perspective. Finally, this first section will end with data analysis requirements in datawarehouse systems.

Concepts and Relations

First of all, it is important to note that the meaning of the words"concept" or "entity" or "object" is not well defined and is not the same in the litterature though they are largely used in datawarehouse systems. In the following we will use them indifferently with the general meaning of complex or structured piece of information. Sometimes a concept will be exactly represented by a table in the relational database, and sometimes it will aggregate information scattered in many tables.
The two first step of the database design process is usually as follows: the domain analysis and its representation in an Entity-Relationship model [1].
During the domain analysis, the relevant concepts of the real world and the relation between these concepts are defined. This distinction between concepts and relations is in some aspects arbitrary and usually depends on the Universe of Discourse (UoD). Briefly, nouns are concepts and verbs are relations. Then, a relationship could be viewed as a concept if the context changes, for instance if a new user have another naming convention. These considerations may seem subtle but it occurs very often that a same query will be conceived in various form by several users.
In other words, the problem is the following: Can a relation be considered as a concept, and *vice versa*, can a concept definition denote the conjunction of other concepts in the same context and then be regarded as a relation ? By example, the concept "employee" in a company can be linked to another employee by the relation "married with". This relation can also be regarded as a concept named "Marriage" that contains references to 2 persons, the wedding date, marriage settlement, etc.
Nowadays, datawarehouse systems show the data model as it was designed for the database implementation. We think that

this representation should not completely depend on the first modeling choice but should be customizable to a new user naming convention. Such a possibility is partially offered with the NIAM methodology [2].
The distinction between Concept and Relation is not the only difficult point for an end-user access. The visualisation of the relations is also important. Indeed, the question is: how to show the links between two objects selected by the user and at which detail level ?

Visualization of links

One of the useful aspects of a database front-end is that end-users do not need to learn SQL for accessing relational database. Many programs automatically construct query joins between tables. In this manner, end-users only have to select the piece of information they want to collect without worrying about the concerned tables and the required SQL joins. However, the invisibility of the links between columns during query definition can lead to misunderstanding of the real meaning of the query.
For instance, let us consider that an end-user wants to know the list of clients that have bought one specific article. He may not see that this query represents the list of clients who sent an invoice that contains an invoice line that refers to the specified article reference.
Firstly, it is important to give the user the ability to visualize the path between client and article in a graphical or a natural language form.
Secondly, the user may want to name "to buy" the relation between clients and articles. In this way, the link path between these two concepts could be recorded in the form of a direct link with the appropriate end-user naming convention.
In this example, the client relational table usually does not contain references to article or even to invoice. Only the invoice table contains a client reference and indirectly an article reference through the invoice-line table. At the conceptual level, the link between client and invoice should be present in both the concept definitions, possibly with different names.

Query management

It is well-known that, for complex nested queries, it can be difficult to verify that the query the user has in mind is correctly constructed and thus translated in SQL. It is important to define a clear representation of the query in a form that allows the user to verify its correctness.
The internal logical query representation should then be viewable in a natural language form. At any moment, the user should be able to see the path link between selected objects or direct link with the appropriate denomination.

Beside query definition consideration, it is interesting to examine optimization needs. One important problem with datawarehouse is that end-users usually may not be aware of the complexity of SQL query he produces and then risks to degradate the database performance. Usually, a datawarehouse system tries to minimize the database access. However datawarehouse users often have to take a previous query result to define another one until they obtain the good information. This interactivity can be very costly for database performance. That is the reason why the query result is stored on the user local station and reporting or analyzing activities are made on the local result. It is then important to store local result in a comprehensive and reusable form. Thus, a datawarehouse system needs optimization during SQL translation and result management.
Once the data retrieved, the user needs to analyse them to extract meaningful information. This data analysis phase is a very important part of datawarehouse.

Data analysis

The datawarehouse market is much concerned with business activities. Usually data analysis consists of filtering and examing the retrieved data from different points of view. In this way much research is pursued on multidimensional structure, named hypercube. Hypercube can, for instance, contain the company turnover for many different years, different clients and different regions. Many datawarehouse systems propose to navigate through this hypercube with automatic aggregate results. Typically, you can see the turnover for a certain year, "drill down" into quarters and then months. The 12 rules of OLAP (OnLine Analytical Processing) - coined by E.F. Codd in a White Paper funded by a vendor [3] - enumerate the minimal requirement for multidimensional systems. Briefly the main requirements concern accessibility, transparency, and navigability into hypercube. The OLAP specification is still rarely integrated in a general datwarehouse system because of the lack of a theoretical foundation.
In a more general view, data analysis consists of building various queries, compare queries between each other and extract meaningful information from the result. It is important to give the user the tools to caracterize the features of the searched information and to compare it to previous result or to target result.
As a matter of fact, such requirements concerning datawarehouse systems imply a strong theoretical base. We will see in the following section a datawarehouse architecture. This framework will propose reasoning capabilities upon a conceptual modeling and query representation system. Natural language will be integrated as an useful interface component.

A FRAMEWORK FOR DATAWAREHOUSE

In this part, we propose some ideas to handle the presented datawarehouse requirements. We have decided to integrate description logics capabilities in the conceptual layer representing relational database information. This may be also useful for natural language processing in a query construction interface. Another interesting consequences of a classification-based system will be semantic query optimization.

Description logics for conceptual modeling

Usually, relational database design is based on the Entity-Relationship model. This model is very suitable for relational database but it lacks of expressivity concerning the kind of relation. Indeed, it is not easy with this model to define is-a relationships or aggregation relationships.

We have decided to describe our model with a frame language associated to terminological reasoning [4] [5]. Briefly, description logics or terminological reasoning associate structural classification capabilities to a logical knowledge represention formalism . Structural classification works on the intensional definition of concepts. The subsumption algorithm realises this classification by inducing generalisation-specialisation relation between concepts. For instance, the concept A is subsumed by the concept B if its intensional definition represents a subclass of the instances defined by the intensional definition of B.

In the context of datawarehouse, classification capabilities may lead to some enhancements. Hence, we would like to handle the possible ambivalence between concept and relation. In this manner, in the client-invoice-article example, we want to define a direct link between client and article as a generalisation (is-a relationship) of the client->invoice->invoice line->article path link. Thus, a subsumption algorithm should be developped to classify such path link.

In the E-R model, certain relationships contain attributes. For instance, Invoice-line may be considered as relationship between Invoice and Article with a Quantity attribute. Actually, this choice depends on naming convention during the design process. If in current use, end-user manipulates a relationship more usually as a concept, the relationship will be objectified and then placed in the concept hierarchy. In this way, we will study logical consequences of such objectification. It is important to note that these considerations take place in the context of natural language. The aim is to allow the system to adapt its own knowledge representation to the user preferences and especially naming convention.

By the way, other particularities from relational database will influence the specification of our description logics-based conceptual modeling formalism. We can quote key values, specific domain restriction, inverse relation, nested queries. Hypercubes, that concern aggregation relationships for numeric values, could also be represented by this formalism [6].

A Natural Language interface

We argue that if natural language is only considered in the restricted context of database with a compact grammar, it can lead to powerful improvements for datawarehouse systems and more generally for database front-ends. Let us present our natural language interface project integrated in a description logics-based conceptual modeling system.

Actually every datawarehouse software system has a natural language layer. This is usually restricted to renaming columns and tables from the database with more comprehensive words for end-users. Several commercial systems propose to name the relation links (that is foreign key columns in relational tables) with verbs and create inverse relation names. We would like to go further and to name a path link like a normal simple link.

For instance, let us consider the following textual query representation:

```
"List of the client
       that have bought the article
                                  named X".
```

If some links have already been defined, a datawarehouse system will be able to calculate the path between client table and article table, that is:

`client->invoice->invoice line>article`.

If the `buy` verb is used for the first time, it would be interesting that the system proposes:

```
"Do you mean that:
a client that have bought an article
  is  a client
        that issued an invoice
           containing an invoice line
              containing an article ?".
```

If the user agrees, the system will record the `buy` direct link between client and article. This new link will be a generalisation of the long path link. Note that the new direct link will not have immediate correspondance with a SQL join but only through its specialized path link.

The natural language interface will be integrated in a multi-modal interface that will involve natural language input and graphical object view for the query construction. The aim is to incrementally build a query in natural language with the assistance of the conceptual model to handle ambiguities and to propose choice lists. At any moment of the query construction, the system should be able to know possible following words and to verify the query consistency.

Query optimization

One useful feature for a modern datawarehouse system concerns query optimization. We have explained above that an end-user does not know the complexity of the SQL query he produces and then risk to degradates the database performance. Furthermore, for data analysis, the user wants to have quick answer in order to easily navigate through data and eventually formulate a new query.

As end-users build query at a conceptual level, the maximum of semantic optimization should be done. If an inconsistency is detected in the query formulation, the system should indicate the source of error and help the user to correct the query. Inconsistency can come from invalid object association or from value constraint violation.

The subsumption algorithm provided by a terminological reasoning system is one possible way to semantically optimize database queries [7]. We consider two queries semantically equivalent if they represent the same result, that is for a relational database, the same set of rows. The result set of the query is then the extensional set of required columns that verify the search conditions. The aim of semantic query optimization is to convert the intensional description of the query into the conceptual model formalism and to simplify the description according to the concept hierarchy. The integrity constraints of each database concept can be propagated to reduce the number of possible instances. If it can be logically deduced that the result set will be empty, the database will not unnecessarily accessed. Furthermore, in the case of inconsistent query, the system may help the user to understand why the result is empty, due to logical inconsistency or lack of information. This correction phase will of course involve the natural language interface.

Another domain of query optimization concerns result local storage. A query can be considered as an intensional concept definition of possible instances. Since we define our conceptual model formalism on description logics, we can define a query in this formalism. In such a way, each query can be structurally compared with each other and then classified in a query hierarchy. Then if a query Q1 subsumes a query Q2, the result set of Q2 will be contained in the result of Q1. Therefore, query classification implies query result classification. In a relational database context, suppose you have locally stored the result of Q1, you do not need, in certain conditions, to submit Q2 to the database since you can restrict your search in the Q1 result table. As a matter of fact, it is possible to construct a local table hierarchy of query results, indexed by the description logics-based query definition. This mecanism may be especially useful for multidimensional data that imply local hypercube storage.

CONCLUSION

In this paper, we have presented some useful requirements for datawarehouse. Indeed, it is important to improve end-user access facilities and to avoid some difficulties during query construction and result exploitation.

We have argued that a database front-end based on a conceptual model with terminological reasoning services can give some enhancements in this direction. Since the modeling phase is compulsory for information system design, it is useful to reuse this model during the database exploitation. We have decided to integrate terminological reasoning in the conceptual modeling formalism. In this way, semantic query optimization and local result storage method will be studied. Furthermore, we would like to give flexibility for the representation of conceptual information with customizable end-user naming convention concerning concepts and relations. As a matter of fact, a natural language interface will directly participate to query definition.

REFERENCES

[1] P. Chen. The Entity-Relationship Model - towards a unified view of data. *ACM Transactions on Database Systems,* 1, 1 (1976), pp.9-36.

[2] G.H.W.M. Bronts, S.J. Brouwer, C.L.J. Martens and H.A. Proper. "A Unifying Object Role Modelling Approach." *Information Systems* Vol.20, n°3 (1995) pp.213-235.

[3] E.F. Codd, S.B. Codd and C.T. Salley. Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate. White Paper funded by Arbor Software. 1993.

[4] T. Kessel, O. Stern and F. Rousselot. "From frames to concepts: building a concept language on a frame-based system." *Int. Workshop on Description Logic (DL-95)* (Rome, 1995) pp. 140-142.

[5] P. Bresciani. "Querying Databases from Description Logics." *KI'95 Workshop in KRDB'95* (Bielefeld, Germany, 1995) pp.1-4.

[6] S. Bergamaschi and C. Sartori. "On Taxonomic Reasoning in Conceptual Design." *ACM Transactions on Database Systems* Vol.17, n°3 (1992) pp.385-422.

[7] D. Benevenato, S. Bergamaschi, S. Lodi and C. Sartori. "Using Subsumption in Semantic Query Optimization." (CIOC-CNR, Bologna, Italia, 1993).