ODMQL: OBJECT DATA MINING QUERY LANGUAGE

Souheir A. Fouad, Amani A. Saad, Mohamed G. Elfeky*

Computer Science and Automatic Control Department
Faculty of Engineering, Alexandria University
Alexandria 21544, Egypt.

Data mining is the discovery of knowledge and useful information from the large amounts of data stored in databases. The emerging data mining tools and systems lead to the demand of a powerful data mining query language. The concepts of such a language for relational databases are referenced herein. With the increasing popularity of object-oriented databases, it is important to design a data mining query language for such databases. The main objective of this paper is to propose an Object Data Mining Query Language (ODMQL) for object-oriented databases as an extension to the Object Query Language (OQL) proposed by the Object Data Management Group (ODMG) as a standard query language for object-oriented databases. The proposed language is implemented as a feature of the ALEX Object-Oriented Database Management System which is a continuous project serving research areas related to Object-Oriented Databases.

يعد التنقيب عن البيانات وسيلة لأكتشاف المعارف والمعلومات النافعة من البيانات ضخمة الحجم المخزنة فى قواعد البيانات. ولقد ادى ظهور أدوات وأنظمة خاصة بعمليات التنقيب عن البيانات الى الحاجة الى لغة استفسار قوية لاداء هذه العمليات ومع زيادة انتشار استخدام قواعد البيانات الشيئية، بات من الضرورى دراسة التنقيب عن البيانات فى هذه القواعد ومن ثم تصميم لغة استفسار خاصة بالتنقيب عن البيانات فيها.

والغرض الاساسى من هذا البحث هو اقتراح لغة استفسار ODMQL للتنقيب عن البيانات فى قواعد البيانات الشيئية كامتداد للغة الاستفسار الشيئية OQL المقترحة كلغة استفسار قياسية لقواعد البيانات الشيئية بواسطة مجموعة ادارة البيانات الشيئية ODMG. وقد تم تصميم وتنفيذ مترجم لهذه اللغة ODMQL وتجربتها فى نظام ادارة البيانات الشيئية ALEX والذى يعد مشروعا بحثيا متواصلا الى خدمة الاغراض البحثية المتعلقة بقواعد البيانات الشيئية.

وتتضمن هذه الورقة البحثية المقدمة اللازمة للبحث ثم تتعرض لخصائص اللغة المقترحة وتشرح تركيباتها المختلفة وامثلة لاستخدامها فى استنباط الانماط التى تمثلها البيانات المخزنة ثم تتعرض لتصميم وتنفيذ المترجم لهذه اللغة.

Keywords: Knowledge Discovery, in Object-Oriented Databases, Data Mining Query Language.

INTRODUCTION

Data Mining means the discovery of knowledge and useful information from the large amounts of data stored in databases [1]. There is a lot of research that has been conducted on data mining in relational databases to mine a specific kind of knowledge [2-7]. Also, there are some data mining experimental systems that have been developed for relational databases, such as DBMiner [8], Explora [9], MineSet [10], Quest [11], etc. The objective of such systems is to mine different kinds of knowledge by offering isolated discovery features. Such systems cannot be embedded into a large application and typically offer just one knowledge discovery feature [12]. The ad hoc nature of knowledge discovery queries in large applications needs an efficient query language much more general than SQL, and this query language is called Data Mining Query Language (DMQL). An example of such language in relational databases is found in Reference 13.

Although the wide variety of advanced database systems relying deeply on the object-oriented data model, there is no data mining query language for object-oriented databases. This motivates us to propose such a language in this paper.

The rest of this paper is organized as follows. The next section describes the design principles of object data mining query languages. Then, we introduce the main

*Currently, Ph.D. student , Computer Science Department, Purdue University

Alexandria Engineering Journal. Vol. 39, (2000) No. 1, 87-96.

features of the proposed language, followed by a formal and complete definition of the language. Next we give a large number of examples serving a wide variety of data mining requests. Finally, we discusses briefly the implementation of the language. The last section summarizes the paper and presents different work to be done in the future.

PRINCIPLES

The definition of data mining may imply that the two terms, Data Mining and Knowledge Discovery, have the same meaning. However, there are some trials to distinguish between those two terms since the first international KDD conference in Montreal in 1995. It was proposed that the term Knowledge Discovery in Databases (KDD) refers to the overall process of discovering useful knowledge from databases while Data Mining refers to a particular step in this process (14). This step considers specific algorithms for extracting specified patterns from data. The additional steps in the KDD process, which are essential before the step of data mining, are data selection, data preparation and data cleaning. Also, there is an additional step after data mining. That is the proper interpretation of the results of data mining algorithms.

Data Selection means to select the set of data in relevance to the knowledge discovery process. *Data Preparation* prepares the background knowledge helping in the data mining step. *Data Cleaning* applies some basic operations, such as the removal of noise and handling missing data fields, on data. *Interpretation* of the mined patterns involves the proper representation of the mined patterns to what may mean knowledge. It may involve visualization and/or evaluation.

Hence, the main principle in designing a data mining query language is to support the specifications of four major primitives:

1. the set of data in relevance to the knowledge discovery process,
2. the background knowledge,
3. the justification of the interestingness of the knowledge (thresholds), and
4. the kind of knowledge to be discovered.

The first primitive can be specified in a way similar to that of an object query used to retrieve a set of objects from the database. The second primitive is a set of concept hierarchies which assist the data mining process. The third primitive can be specified through a set of different mining thresholds depending on the kind of knowledge to be discovered, that is the fourth primitive, that can be specified by stating explicitly the type of knowledge to be mined in the current knowledge discovery process. In the next sub-sections, each one of these primitives, except the first, will be discussed in more detail.

Concept Hierarchies

A concept hierarchy defines a sequence of mappings from a set of concepts to their higher-level correspondences. Concept hierarchies represent necessary background knowledge to control the generalization process that is a preliminary step in most data mining algorithms. Generalization of an attribute means to replace its value by a higher one based on a concept hierarchy tree. For example, a person's address can be generalized from a detailed address, such as the street, into a higher leveled one, such as a district, a city, a country, etc. based on the concept hierarchy tree shown in Figure 1. Hence, generalization of an object means to generalize one or more of its properties using a pre-specified concept hierarchy tree for each property. The trees shown in Figure 1 show different levels of concepts. Note that the word *"ANY"* is a reserved word for the root of the tree. Using concept hierarchies, the knowledge mined from the database can be represented in terms of generalized concepts and stated in a simple and explicit form.

Rules

The extracted patterns may be represented in many forms according to the

data mining method used. Some of those forms are classification trees, neural networks, multidimensional regression, or more generally *rules*. An example of such rules represented in first-order predicate calculus is

x.diagnosis = "Heart" And x.sex = "Male"==>x.age>50[1200 , 0.70]

indicating that there are 1200 male persons with heart attack and that 70% of them are over 50.

A rule is composed of a Body, a Consequent, a Support, and a Confidence. The *Body* of the rule is the part before the implication operator representing the examined data. The *Consequent* is what follows the implication operator representing a discovered property for the examined data. The *Support* is the value representing the number of records in the whole data satisfying the *body* clause. The *Confidence* is the value representing the percentage of the records satisfying both the *body* and the *consequent* clauses to those satisfying only the *body* clause.

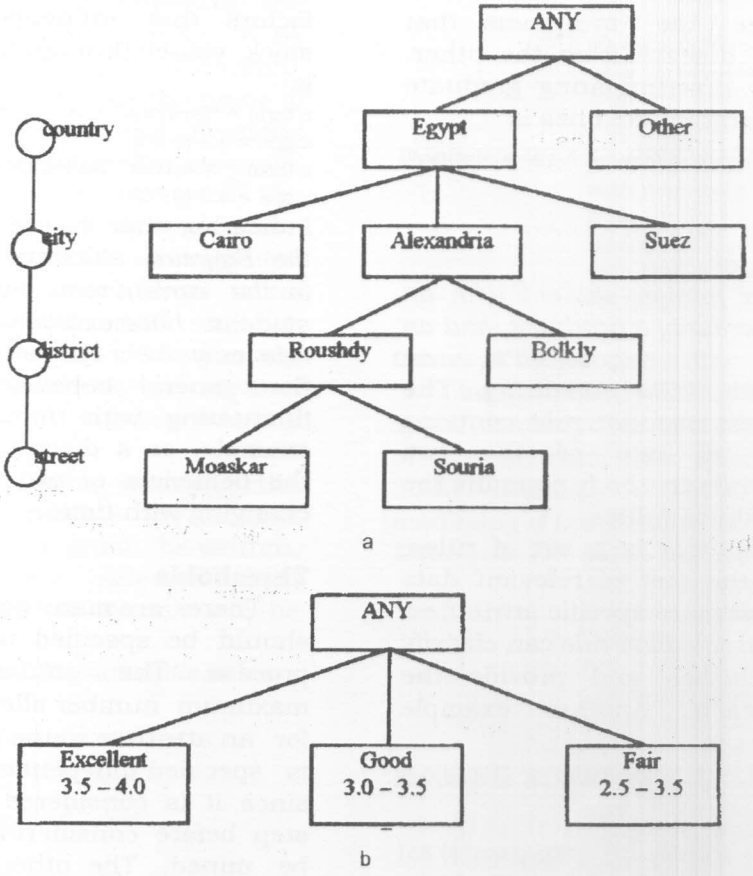

Figure 1 (a): A concept hierarchy tree for the attribute "address"; (b) A concept hierarchy tree for the attribute "GPA"

There are different types of knowledge to be discovered. Hence, there are so many types of rules including Characteristic, Discriminant, Classification, Association, Data Evolution, and Deviation rules according to the data mining method applied.

A *characteristic rule* is an assertion that characterizes the concepts satisfied by all or most of the objects in the set of relevant data. For example, the symptoms of a specific disease can be summarized by a *characteristic rule*. Another example characterizing graduate students is

x.status="Graduate"==>

(x.nationality = "Egyptian" And x.gpa>3.5 [0.75])
Or (x.nationality = "Foreign" And x.gpa > 3.0 [0.25])

indicating that *a graduate student is either Egyptian with an excellent GPA (with 75% probability) or foreign with a good GPA (with 25% probability).* The *body* of a *characteristic rule* contains the specification of the set of relevant data being characterized, while the *consequent* contains the characterizing attribute values.

A *discriminant rule* is an assertion that discriminates concepts of two contrasting sets of data. For example, to distinguish one disease from another, a *discriminant rule* should summarize the symptoms that discriminate this disease from the other. Another example discriminating graduate students from undergraduate ones is

x.nationality = "Foreign" And x.gpa > 3.5 ==> x.status = "Graduate" [1.00]

x.nationality = "Egyptian" And x.gpa > 3.0 ==>
x.status = "Under" [0.90]

indicating that *a foreign student with an excellent GPA is certainly a graduate, and an Egyptian student with a good GPA is an undergraduate with 90% probability.* The *consequent* of a *discriminant rule* contains the specification of one of the two contrasting sets, while the *body* contains the discriminating attribute values.

A *classification rule* is a set of rules, which classifies the set of relevant data according to one or more specific attributes. For example, a *classification rule* can classify diseases into classes and provide the symptoms of each. Another example classifying students is

x.status = "Graduate" ==> x.nationality = "Egyptian" [0.60]
Or x.nationality = "Foreign" [0.40]
x.status = "Under" ==> x.nationality = "Egyptian" [0.85]
Or x.nationality = "Foreign" [0.15]

indicating that *regarding the status and nationality, students are either graduates (60% Egyptians and 40% foreign) or undergraduates (85% Egyptians and 15% foreign).* Each distinct value of the attribute, according to which the data is classified, must appear in the *consequent* of a rule. The *body* contains the classifying attribute values for each distinct value.

An *association rule* describes association relationships among the set of relevant data. For example, an *association rule* may discover a set of symptoms frequently occurring together. Each attribute of the relevant set of data being examined can appear either in the *body* or in the *consequent* of the *association rule* according to its role in the rule.

A *Data Evolution Rule* reflects the general evolution behavior of the set of relevant data. Clearly, this kind of rules is valid only in time-related (temporal) data. For example, a *data evolution rule* can describe the major factors that influence the fluctuations of stock values through time. Another example is

x.term = "Previous" And x.nationality = "Egyptian" ==>
x.gpa > 3.5 [0.80]
x.term = "Current" And x.nationality = "Egyptian" ==>
x.gpa > 3.5 [0.70]

indicating that *in the previous term, 80% of the Egyptian students had excellent GPA, but in the current term, only 70% of the Egyptian students have excellent GPA.* A *data evolution rule* may be a characteristic rule describing the general behavior of a set of data fluctuating with time, just like the above example, or a discriminant rule comparing the behaviors of two different sets of data changing with time.

Thresholds

There are many kinds of thresholds that should be specified to control the mining process. The *attribute threshold* is the maximum number allowed of distinct values for an attribute in the generalized objects. It is specified independent of the kind of rules since it is considered in the generalization step before considering the kind of rules to be mined. The other kinds of thresholds depend on the specified type of rules being mined. For example, mining *association rules* should specify a *support threshold* that is the minimum support value of a rule, and a *confidence threshold* that is the minimum confidence value of a rule. Also, mining *classification rules* should specify a *classification threshold* such that further classification on a set of classified objects may become unnecessary if a substantial

portion (no less than the specified threshold) of the classified objects belong to the same class.

FEATURES

The proposed ODMQL allows the user (data miner) to write data mining queries and specify each primitive presented above in a an OQL-like syntax. The following sub-Sections discuss the main features of the proposed language.

Kind of Rules

The proposed ODMQL supports the specification of the kind of rules to be extracted in the current query. Certainly, the specification should include the name of this kind and some other specifications according to the specified kind. For example, mining *discriminant rules* should specify the two contrasting classes. The kinds of rules supported are *characteristic*, *discriminant*, *classification* and *association*. There is no need to support explicitly *data evolution* rules since they can be considered as any other kind of rules with the specification of one or more time-related attributes in any part of the data mining query.

Note that in the following examples, the **bold** and *italic* words indicate keywords of the language. **Bold** ones must be written, while *italic* ones are selected from alternatives. The complete syntax will be discussed later.

Example 1:
mine for *Characteristic* **rules**

Example 2:
mine for *Discriminant* **rules**
contrasting x.status = "Graduate"
with x.status = "Under"

Example 3:
mine for *Classification* **rules**
according to x.term

Relevant Data

The set of relevant data is specified in the proposed ODMQL just like the way in OQL replacing the word "***select***" with "***with relevance to***" to follow the meaning of the data mining query.

Example 4:
mine for *Characteristic* **rules**
with relevance to x.nationality , x.gpa
from Student x
where x.status = "Graduate"

Thresholds

The proposed ODMQL supports the specification of the previously discussed thresholds. Note that each attribute could have a different *attribute threshold*. Mentioning just one *attribute threshold* means that this threshold is applied for all the attributes.

Example 5:
mine for *Association* **rules**
with relevance to x.store, x.product, x.date
from Sales x
where x.date.year = "1998"
with thresholds (*Attribute* = *Support* = 0.35 , *Confidence* = 0.25)

Example 6:
mine for *Characteristic* **rules**
with relevance to x.nationality , x.gpa
from Student x
where x.status = "Graduate"
with thresholds (*Attribute*=3, *Attribute* = 4)

Concept Hierarchies

The proposed ODMQL allows the user to specify concept hierarchies for the attributes of the schema. The specification includes both defining a new hierarchy, and modifying a pre-defined one.

Example 7:
define hierarchy for major :
ANY -> { Science, Art } ,
Science->{Biology, Chemistry, Computer } ,
Art -> { Music, Literature , ... },
Chemistry->{Analytical,Biochemistry,...},
Music -> { Rock, Pop, Arabic, ... }

Example 8:
define hierarchy for gpa:
ANY -> { Excellent , Good , Fair } ,
Excellent -> [3.5 .. 4.0] ,
Good -> [3.0 .. 3.5] ,
Fair -> [2.5 .. 3.5]

Example 9:
modify hierarchy for gpa :
delete ANY -> Fair ,
insert ANY -> { Average , Poor } ,
insert Average -> [1.5 .. 2.5] ,
insert Poor -> [0.0 .. 1.5] ,
update Good -> [2.5 .. 3.5]

SYNTAX

The proposed ODMQL syntax is given in an extended BNF grammar using the following notations:

{symbol} represents zero or more occurrences of this symbol.
[symbol] represents zero or one occurrence of this symbol.
keyword represents a terminal of the grammar.
<symbol> represents a non-terminal of the grammar.
symbol1 | symbol2 represents either the first symbol or the second.

Grammar

```
<ODMQL> ::=
        <data_mining_query> |
      <concept_hierarchy_query>


<data_mining_query> ::=
      mine for <rule_specification>
      with relevance to <projection_attributes>
      from <variable_declaration> {,
<variable_declaration>}
      [where <query>]
      [with threshold[s] ( <threshold> {, <threshold>}
)]

<rule_specification> ::=
    Characteristic rules|Association rules |
    Discriminant rules contrasting <query> with
<query> |
    Classification rules according to
<projection_attributes>

<threshold> ::= <threshold_type> = <numerical_value>

<threshold_type> ::=
        Attribute | Support | Confidence |
      Classification

<concept_hierarchy_query> ::=
                  <define_hierarchy> |
                <modify_hierarchy>

<define_hierarchy> ::=
            define hierarchy for <projection> :
            ANY -> <concept _set> {,
<concept_definition>}

<modify_hierarchy> ::= modify hierarchy for
<projection> :
                    <modification> {, <modification>}

<modification> ::= delete <concept> -> <concept> |
                   insert <concept_definition> |
                   update <concept_definition>

<concept_definition> ::= <concept> -> <concept_set>

<concept_set> ::= { <concept> {, <concept>} } |
                  [ <numerical_value> ..
        <numerical_value> ]

<concept> ::= ANY | <string_literal>

<numerical_value> ::= <integer_literal> | <float_literal>
```

Note that the non-terminal <projection> indicates either an attribute or a member, qualified by the type name or not qualified at all. In the previous grammar, any non-terminal without definition represents a non-terminal of the OQL BNF presented in Reference 15.

EXAMPLES

In this section, we try to develop extensive examples to show the various capabilities of the proposed ODMQL. The examples presented here are grouped into two categories, one for the *concept hierarchy* queries, and the other for the *data mining* queries.

Concept Hierarchy Queries

Example 1 shows how to define the concept hierarchy shown in Figure 1-A.

Example 1:

```
define hierarchy for address :
ANY -> { Egypt , Other } ,
Egypt -> { Cairo , Alex , Suez } ,
Alex -> { Roushdy , Bolkly } ,
Roushdy -> { Moaskar , Souria }
```

Example 2 shows how to define a concept hierarchy for a numerical attribute. Example 3 shows how to modify that previously created concept hierarchy by insertion of new concepts, deletion of pre-existing ones, and updating the values of pre-existing ones. Figure 2 shows the concept hierarchy tree before and after the modification.

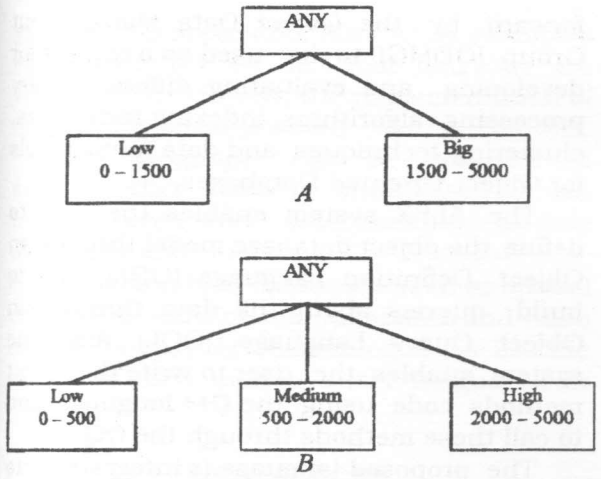

Figure 2 Concept hierarchy tree for the attribute "salary" : (A) Before modification, (B): After modification

Example 2:
define hierarchy for salary :
ANY -> { Low , Big } ,
Low -> [0 .. 1500] ,
Big -> [1500 .. 5000]

Example 3:
modify hierarchy for salary :
insert ANY -> { Medium , High },
delete ANY -> Big ,
update Low -> [0 .. 500] ,
insert Medium -> [500 .. 2000],
insert High -> [2000 .. 5000]

Example 4 shows how to define a concept hierarchy for the attribute "*month*". The concept hierarchy tree is shown in Figure 3.

Example 4:
define hierarchy for month :
ANY -> { Half1 , Half2 } ,
Half1 -> { Quarter1 , Quarter2 } ,
Half2 -> { Quarter3 , Quarter4 } ,
Quarter1 -> [1 .. 3] ,
Quarter2 -> [4 .. 6] ,
Quarter3 -> [7 .. 9] ,
Quarter4 -> [10 .. 12]

Data Mining Queries

Example 5 shows a query to mine the *characteristic rules* related to the attributes "*nationality*" and "*gpa*" of the graduate students. Each attribute has its own *attribute threshold*.

Example 5: **mine for Characteristic rules**
with relevance to x.nationality ,
x.gpa
from Student x
where x.status = "Graduate"
with threshold (**Attribute** = 3
Attribute = 4)

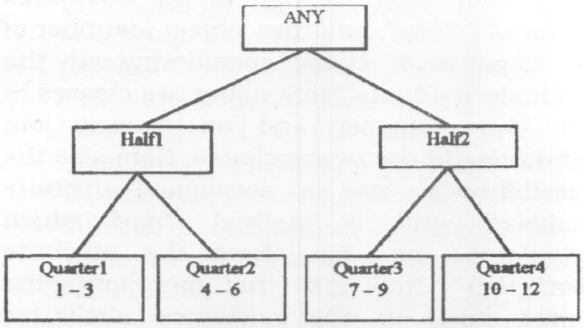

Figure 3 A concept hierarchy tree for the attribute "month"

Example 6 shows a query to mine the *discriminant rules* contrasting graduate students with undergraduates in the computer science department according to the attributes "*nationality*" and "*gpa*".

Example 6:
mine for Discriminant rules
contrasting s.status = "Graduate" **with**
s.status = "Under"
with relevance to s.nationality , s.gpa
from Student s
where s.into.name = "cs"
with threshold (**Attribute** = 5 ,
Confidence = 0.90)

Example 7 shows a query to mine the *classification rules*, related to the attribute "*nationality*", which classifies the graduate students according to their departments. Note using a path expression *s.into.id* in the *according to* clause since it is supported in the object-oriented data model. This path expression means the *department id* of this *student* since *s.into* refers to an object of the class *department* and *id* is an attribute of that class.

Example 7
mine for Classification rules
according to s.into.id
with relevance to s.nationality
from Student s
where s.status = "Graduate"
with thresholds (**Attribute** = 4 , **Classification** = 0.85)

Example 8 shows a query to mine the *association rules*, related to the attributes "*hobbies*", "*age*" and the object identifier of the department class, considering only the graduate students. Note using two classes in the *from* clause and an explicit join condition in the *where* clause. Note also the possibility to use a set-valued attribute "*hobbies*" and a method "*age*" which calculates the age from the attribute "*birth_date*". Note also that mentioning the object itself in the relevance attributes means the object identifier that is generalized using the inheritance hierarchy [1].

Example 8:

mine for Association rules
with relevance to s.hobbies , d , s.age ()
from Student s , Department d
where (s.status = "Graduate") And
(s.into.id = d.id)
with thresholds (**Attribute** = 4,
Support = 0.75 ,
Confidence = 0.85)

Example 9 shows a query to mine the *discriminant rules* contrasting the sales performed in 1998 with those performed in 1997 according to the attributes "*store*" and "*product*". Those rules may be considered as *data evolution rules* comparing the behaviors of some data in two contrasting times.

Example 9:

mine for Discriminant rules
contrasting x.date.year = "1998" **with**
x.date.year = "1997"
with relevance to x.store , x.product
from Sales x
with threshold (**Attribute** = 5)

IMPLEMENTATION

The proposed Object Data Mining Query Language (ODMQL) is a simple language used to mine knowledge from object-oriented databases. Integration of such language into the ALEX system [15] makes it possible to examine this language and extract knowledge from object-oriented databases. ALEX is a continuous research project whose objective is to build an OODBMS that follows the standards put forward by the Object Data Management Group (ODMG) to be used as a testbed for developing and evaluating different query processing algorithms, indexing techniques, clustering techniques, and data mining tools for Object-Oriented Databases.

The ALEX system enables the user to define the object database model through an Object Definition Language (ODL), and to build queries about the data through an Object Query Language (OQL). Also, the system enables the user to write the object methods code using the **C++** language, and to call these methods through the OQL.

The proposed language is integrated into ALEX by implementing an ODMQL processor that executes the data mining statements as shown in Figure 4. The ODMQL processor includes a parser that parses the statements, a concept hierarchy manager that manipulates the concept hierarchies, and data mining techniques to mine different kinds of rules. The ODMQL processor uses the OQL processor to retrieve the set of relevant objects that will participate in the current data mining process.

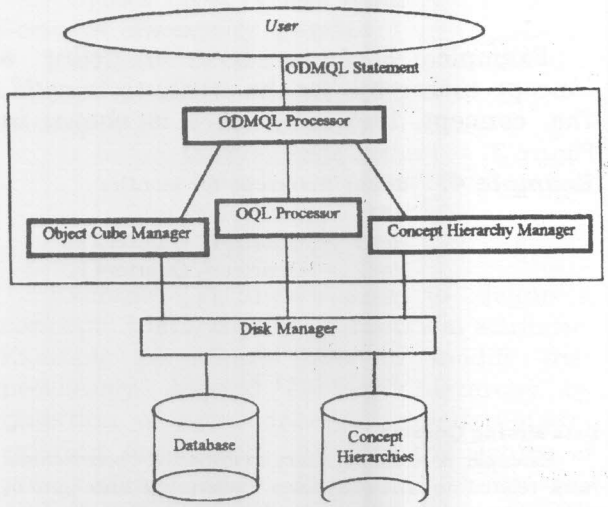

Figure 4 Data mining subsystem in ALEX.

CONCLUSION AND FUTURE WORK

In this paper, a new query language for mining object-oriented databases is proposed. The concepts and features of this language are outlined, and its syntax

grammar is presented. Also, some examples of queries written in that language are encountered. Finally, the implementation of this language is discussed.

This language can be extended to support other kinds of knowledge that can be discovered from databases such as data deviations and clusters.

Other kinds of databases such as spatial databases, multimedia databases and video databases may be examined to discuss the design of data mining query languages for them.

REFERENCES

1. J. Han, S. Nishio, H. Kawano, and W. Wang, "Generalized-Based Data Mining in Object-Oriented Databases Using an Object Cube Model", Data and Knowledge Engineering, Vol. 25, No. 1-2, pp. 55-97, (1998).
2. R. Agrawal, M. Mehta, J. Shafer, R. Srikant, A. Arning and T. Bollinger. "The Quest Data Mining System", Proceedings of 1996 International Conference on Data Mining and Knowledge Discovery (KDD'96), pp. 244-249, Portland, Oregon, August (1996).
3. J. Han and Y. Fu, "Discovery of Multiple-Level Association Rules from Large Databases", Proceedings of 1995 International Conference on Very Large Databases, pp. 420-431, Zurich, Switzerland, September (1995).
4. M. Mehta, R. Agrawal, and J. Rissanen, "SLIQ: A Fast Scalable Classifier for Data Mining", Proceedings of 1996 International Conference on Extending Database Technology, Avignon, France, March (1996).
5. R. Srikant and R. Agrawal. Mining Generalized Association Rules. In Proceedings of 1995 International Conference of Very Large Databases, Zurich, Switzerland, pp. 407-419, September (1995).
6. R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases", Proceedings of 1993 ACM-SIGMOD International Conference on Management of Data, pp. 207-216, May (1993).
7. T. Zhang, R. Ramakrishnan and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases", Proceedings of 1996 ACM-SIGMOD International Conference on Management of Data, Montreal, Canada, June (1996).
8. J. Han, J. Chiang, S. Chee, J. Chen, Q. Chen, S. Cheng, W. Gong, M. Kamber, G. Liu, K. Koperski, Y. Lu, N. Stefanovic, L. Winstone, B. Xia, O.R. Zaiane, S. Zhang, and H. Zhu. "DBMiner: A System for Data Mining in Relational Databases and Data Warehouses", Proceedings of CASCON'97, Toronto, Canada, November (1997).
9. W. Klösgen, "Explora: A Multipattern and Multistrategy Discovery Assistant", U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pages 249-271. AAAI / MIT Press, (1996).
10. C. Brunk, J. Kelly, and R. Kohavi. "MineSet: An Integrated System for Data Mining", Proceedings of Third International Conference on Data Mining and Knowledge Discovery (KDD'97), pp. 135-138, Newport Beach, California, August (1997).
11. R. Agrawal, M. Mehta, J. Shafer, R. Srikant, A. Arning and T. Bollinger. "The Quest Data Mining System", Proceedings of 1996 International Conference on Data Mining and Knowledge Discovery (KDD'96), pp. 244-249, Portland, Oregon, August (1996).
12. T. Imielinski and H. Mannila, "A Database Perspective on Knowledge Discovery", Communications of the ACM, No. 11, pp. 58-64, (1996).
13. J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaiane, "DMQL: A Data Mining Query Language for Relational Databases", SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge

16. S.A. Fouad, A.A. Saad, and M.G. Elfeky, "ALEX Object-Oriented Database Management System", Published in ICCTA'99 International Conference on Computers: Theory and Applications, Alexandria, Egypt, August (1999).

Received September 30. 1999
Accepted December 14. 1999