*J. King Saud Univ.*, Vol. 10, *Comp. & Info. Sci.*, pp. 15-25 (A.H. 1418/1998)

# Knowledge Discovery in Databases: A Query-Guided Approach

**Saud S. Al-Mathami**
*Computer Science Department, College of Computer & Information Sciences*
*King Saud University, Riyadh, Saudi Arabia*

(Received 11 November 1995, accepted for publication 29 May 1996)

**Abstract.** The paper identifies a novel approach to mine knowledge from data, where answers returned to queries issued by a user are used to guide supervised knowledge discovery in data. This approach differs from previous, unsupervised approaches which use the query content and structure only. Examples are presented to demonstrate the feasibility of the approach, called query guided knowledge discovery in databases (QGKDD).

## 1. Introduction

As a result of growing interest in the area of knowledge discovery in databases (KDD), the relationship between machine learning techniques and their use with database systems is coming under increasing scrutiny. KDD is the process of applying statistical, machine learning and other techniques to conventional database systems [3]. The data is typically excavated or mined to produce structured representations (e.g. identification trees, production rules) of the domain [1], [7-8], [10]. While such techniques are well-understood, their application to large databases has led to problems of inefficiency and so to the search for guided KDD, where such guidance can be based on prior knowledge or on applying data mining techniques to 'windows' (subsets) of data [11]. The problem of handling subsequent inconsistencies when guidance has been used is a major one [6]. The inconsistencies arise when the maintenance and the updating of knowledge base which is still an open research area are intended to take *conflicting* and *exceptional* information into account as it arises. Most knowledge structures are monotonic, where a conclusion $Q$ from a set of knowledge premises $K$ is guaranteed to hold for every superset of $K$. A significant amount of knowledge is not monotonic. If $K_1$ is {Tweety is a bird, birds fly} and $Q_1$ is 'Tweety flies', and if $K_2$ is the superset {Tweety is a bird, birds fly, Tweety is a penguin, penguins are birds, penguins don't fly}, then $Q_1$ is no longer guaranteed to hold in the face of $Q_2$ 'Tweety doesn't fly'.[1] When an inconsistency occurs, rule-based truth maintenance systems may not effectively and tractably provide intuitively correct updates to the knowledge base. Also, unless some preference rules are built in, the system will not know which of the two conflicting conclusions to prune. Preference-based approaches where, for example, each superset is given its own level and higher level conclusions are to be preferred (to represent the intuition that a superset consists of later information which fills in gaps in earlier information) can require that every time a new fact is known, a new superset with its own unique level be generated. There may be as many levels as there may be as many levels as there are facts, for instance.

For this reason. exceptions and conflicts are not generally integrated into the knowledge structure but may be kept in separate tables pointed to by the appropriate rule in the rule-base. if the number of exceptions and conflicts grows sufficiently large, they can be appended to the original data and a new knowledge structure with associated rule-base generated. Windowing techniques in the Inductive Deductive System (ID4) involve adding exceptions to the original data and generating a new tree [11]. ID4 prunes subtrees of the knowledge structure which appear to be incorrect and adds whole new subtrees as new instance appear, no matter whether the information conflicts with existing information or not. The question addressed here is whether a different sort of knowledge structure can be generated such that conflicting and exceptional information can be integrated directly into the knowledge structure as they arise (revisable knowledge discovery in databases-RKDD), rather than be stored separately for later recompilation.

KDD techniques can be used to extract knowledge for *managing* the data. For instance, the application of KDD techniques on a large database can help facilitate the construction of smaller database systems for distribution across a network, where the distribution is based on identified relationships between subsets of attributes.

However, the size of the original database can sometimes prevent the efficient use of KDD techniques for this purpose. One way to overcome this problem is to incorporate machine learning algorithms into a database system for monitoring the queries issued on the system and then using the structured representations generated from the queries to provide information to the database administrator for dynamically modifying the physical and external schemes of the database to achieve improved system performance [4]. Another approach is to by-pass the use of structured representations altogether and to use the queries themselves to formulate knowledge rules [12].

The focus of this paper is on a novel technique for helping database

[1] Note that there is no inconsistency between $k_1$ and $k_2$.

administrators manage a large database system. Whereas the approach of [4] uses the structure of the queries to generate concept hierarchies based on the predicates appearing in the queries, and the approach of [12] uses the conditions of the query directly in derived rules, the approach to be described here goes a significant step further by applying supervised machine learning techniques on the data returned by the queries so that such techniques come up with their own classification of the data which may bear only some correspondence to the original query. Not only does this avoid some of the basic problems which arise when only queries are mined (to be described below), but also it allows a degree of 'grounding' or semantic plausibility for restructuring databases, whereas previous approaches can only appeal to syntactic plausibility. The mining of a large database is now in partnership with the queries issued upon it, and as queries become progressively deeper so does the knowledge extracted from the database. The advantages are that information on how to restructure the database is now solidly based on the actual use made of the database rather than idealized use which, while important for generating initial internal and external schemes, must be modified dynamically in the right direction if user productivity is to be improved.

In the long term the approach to be described here provides a way of integrating databases and Artificial Intelligence (AI) which takes into account the benefits of both technologies without compromising either. It has long been recognized by researchers in the two communities that AI and databases need to be integrated in a manner which is systematic and coherent. Recent evidence indicates that AI techniques can be successfully adopted for aiding the database design process, and further evidence is provided in this paper that AI techniques can significantly aid the database administrator's role and function, especially with regard to fine-tuning of the database system as queries are processed.

## 2. Current Use of Queries in KDD

### 2.1. Concept-based knowledge discovery

Recently, machine learning techniques have been used to build hierarchical structures to be analyzed by the DBA [4]. This approach uses conceptual clustering (un-supervised learning) to construct hierarchical structures from queries. For instance, imagine there are two relations EMP (eno, ename, age, salary, edno) and DEPT (dno, dname, floor) (adapted from [4]), where EMP contains information on employee number, name, age, salary and department number, and DEPT on department number, name and floor located. Using the unsupervised learning algorithm UNIMEM [5] on the following queries [2] :

[2] These examples are adapted from those presented in [4] and are not so detailed as the original examples .

1. **select** *ename*
   **from** EMP, DEPT
   **where** *edno = dno*
2. **select** *ename*
   **from** EMP, DEPT
   **where** *edno = dno* **and** *age = 20* **and** *salary = 200k*
3. **select** *ename*
   **from** EMP, DEPT
   **where** *edno = dno* **and** *age = 20*

The tree as given in Fig.1 is produced. The idea then is to use the information in the concept hierarchy to aid the database administrator in aspects of external and physical schema design (e.g. by creating indices for high level concepts or new external views).

There are a couple of limitations to the approach adopted by [4]. First, the approach uses only the information contained in the query itself and does not take into account the answers returned to the query. This means that two queries which are syntactically different but return the same answers may be distributed across different nodes of the concept hierarchy, although they both refer to the same data. The concept hierarchy therefore is purely *intentional*. If the answers returned to a query can be taken into account when generating a concept hierarchy, then the resulting hierarchy can be both intentionally and *extensionally* grounded.

Secondly, as a result of not taking the answers returned into account, the learning approaches adopted (UNIMEM and COBWEB[2]) are necessarily unsupervised. If the answers returned to a query are tagged as 'positive' and all other samples 'negative', a supervised learning approach can be adopted with the attendant advantages that supervised learning brings (e.g. the use of mathematically rigorous information-theoretic approaches, progressive deepening of the identification tree for samples not correctly classified, more intuitively interpretable trees with clear tests on attribute values). The task of the supervised learning algorithm will then be to identify what distinguishes the positive samples from the negative samples. If the number of negative samples is huge, a random subset of such samples can be used, since what is at issue is what distinguishes the set of positive samples from the (perhaps randomly chosen) negative samples.

### 2.2. Query-based knowledge discovery

The approach of [12] uses a query to distinguish between 'positive' and 'negative' examples, where those examples returned as answers to a user query are considered 'positive' and those which are not 'negative'. If $q$ is a query condition of a query $Q$, and $r_1$ is a clause which is satisfied for $q$ then the rule $q \rightarrow r_1$ is deduced for positive examples. If $q$ is a query condition of the query $Q$ and $r_2$ is a clause not satisfied for

$q$, then the rule $q \wedge r_2 \rightarrow$ is deduced (i.e. no conclusion can be drawn given these conditions). Once rules are deduced for each attribute condition in the query, a further process of 'knowledge harmonization' is required so that only 'useful' rules are considered for database knowledge, where 'useful' is defined as consistent and non-redundant.

The process of constructing a concept hierarchy based on the queries, using a UNIMEM learning approach is shown in fig. 1, (a) The concept hierarchy is empty at the start. (b) When the query '**select** ename **from** EMP, DEPT where *edno = dno'* is encountered, the child *'edno = dno'* is inserted. (c) When the query '**select** ename **from** EMP, DEPT where *edno = dno* and *age = 20* and *salary = 200k'* is encountered, a new child node containing only the features not present in the earlier query is created. (d) When the query '**select** ename **from** EMP, DEPT where *edno = dno* and *age = 20* ' is encountered, the *'age'* and *'salary'* features as split, with '*salary*' features inheriting the '*age*' features. [4] subsequently attach a concept name to each node (e.g. 'employment' for the *'edno = dno'* feature node, 'young employee' for the *'age = 20'* node), thereby claiming that the result is a concept hierarchy, but they do not make it clear where these concept names come from. Also, the approach of [4] involves inserting confidence measures into the nodes, details of which are ignored here. Finally, [4] also demonstrate the construction of a concept hierarchy using a COBWEB approach, details of which are also ignored here.

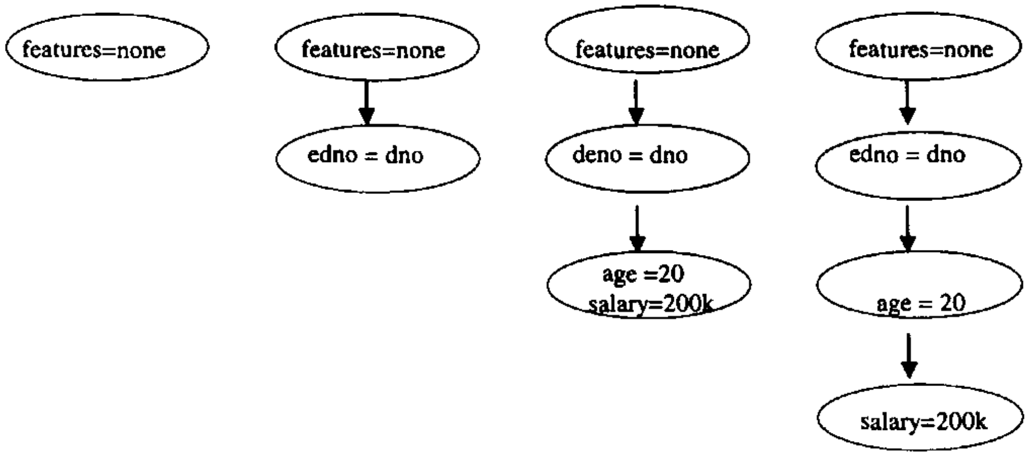

**Fig. 1. The process of constructing a concept hierarchy based on the queries using UNIMEM.**

In [12] the claim is made that using structural information (such as provided by ID3) is not efficient when data are inconclusive or when there are fewer positive data and many more negative data. This claim, which is not further substantiated in [12], is used to support the idea of moving directly from query conditions to rules based on resulting positive and negative samples. But it is not clear how the approach described in [12] would deal with two different queries returning the same data (positive samples), and how knowledge harmonization would identify deeper relationships among the data than those identified by query conditions.

## 3. Query Guided Knowledge Discovery

### 3.1. Introduction to QGKDD

The approach to be described here follows a third route. Rather than knowledge discovery in data being based on user queries, such KDD is guided by user queries-query-guided KDD (QGKDD). The information returned to a query is viewed as positive examples of a 'target' concept, while the information not returned to a query is viewed as negative examples of a 'target' concept. Supervised learning algorithms (ID3) are then run on the data, with the task of identifying which attributes in the dataset as a whole are best for distinguishing the positive and negative samples, irrespective of the query conditions. One would expect to find some correspondence between query conditions and the way a supervised learning algorithm classifies by means of a decision tree the positive and negative samples, but it is the task of the learning algorithms to discover this correspondence. In the worst case, no correspondence may exist.

### 3.2. Example 1

Consider the toy relational dataset provided in Table 1, and imagine that the following SQL- like query is posed:

**select***
**from** Zoo
**Where** legs = 4 **and** aquatic = no

**Table 1. An example zoo dataset, consisting of 16 tuples and 6 attributes**

| Animal | Domestic | Airborne | Aquatic | Legs | Catsize |
|---|---|---|---|---|---|
| Bear | no | no | no | 4 | yes |
| Crow | no | yes | no | 2 | no |
| Chicken | yes | yes | no | 2 | no |
| Deer | no | no | no | 4 | yes |
| Dove | yes | yes | no | 2 | no |
| Duck | no | yes | yes | 2 | no |
| Frog | no | no | yes | 4 | no |
| Gull | no | yes | yes | 2 | no |
| Hawk | no | yes | no | 2 | no |
| Lobster | no | no | yes | 6 | no |
| Moth | no | yes | no | 6 | no |
| Newt | no | no | yes | 4 | no |
| Oryx | no | no | no | 4 | yes |
| Penguin | no | no | yes | 2 | yes |
| Seasnake | no | no | yes | 0 | no |
| Swan | no | yes | yes | 2 | yes |

The tupelos satisfying this query are those for bear, deer and oryx. What then happens is that an additional temporary attribute, *class*, is appended to the relation, with *yes* values if the corresponding tupelo satisfies the query, otherwise *no* (Table 2). Running ID3 on the resulting dataset produces the identification tree as provided in Fig. 2. From this tree, the following rules can be derived:

1. **if** catsize = *no* **then** animal is crow or chicken or dove or duck or frog or gull or hawk or lobster or moth or newt or seasnake;
2. **if** *catsize* = *yes* and *aquatic* = *no* **then** animal is bear or deer or oryx;
3. **if** *catsize* = *yes* and *aquatic* = *yes* *then* animal is penguin or swan.

These rules can be optimized if necessary. The important point is that from the query:

**select** *
**from** Zoo
**where** *legs* = *4* **and** *aquatic* = *no*

a different way has been found to identify the same examples. The user, when presented with this identification tree, can choose to associate a concept with the returned class of examples (e.g. 'non-aquatic but catsized'). The query's original reference to legs has been shown not to be necessary for identifying which samples are to be returned.

## 3.3. Example 2

Now consider the following query on the Zoo dataset:

**select** *
**from** Zoo
**Where** *legs* = *2*

This produces the relation given in Table 3 which, when fed to ID3, produces the identification tree given in Fig. 3.

Here there is a direct correspondence between the conditions of the query and the resulting identification tree, in that ID3 has found that distinguishing among the values of the 'legs' attribute leads to the most efficient classification. This attribute is also the one chosen by the user in the second query. However, what becomes clear in this example is that all the birds fall in the two-lecategory. The user, when presented with the full identification tree, can then associate the concept 'bird' with the induced rule: 'If the creature has two legs, then it is a bird' (given the current state of the database).

**Table 2. The previous dataset with the new attribute class appended, with values signifying whether the tuple satisfies the query. Appending such a class and its values allows supervised learning algorithms to be run on the dataset**

| Animal | Domestic | Airborne | Aquatic | Legs | Catsize | Class |
|---|---|---|---|---|---|---|
| Bear | no | no | no | 4 | yes | yes |
| Crow | no | yes | no | 2 | no | no |
| Chicken | yes | yes | no | 2 | no | no |
| Deer | no | no | no | 4 | yes | yes |
| Dove | yes | yes | no | 2 | no | no |
| Duck | no | yes | yes | 2 | no | no |
| Frog | no | no | yes | 4 | no | no |
| Gull | no | yes | yes | 2 | no | no |
| Hawk | no | yes | no | 2 | no | no |
| Lobster | no | no | yes | 6 | no | no |
| Moth | no | yes | no | 6 | no | no |
| Newt | no | no | yes | 4 | no | no |
| Oryx | no | no | no | 4 | yes | yes |
| Penguin | no | no | yes | 2 | yes | no |
| Seasnake | no | no | yes | 0 | no | no |
| Swan | no | yes | yes | 2 | yes | no |

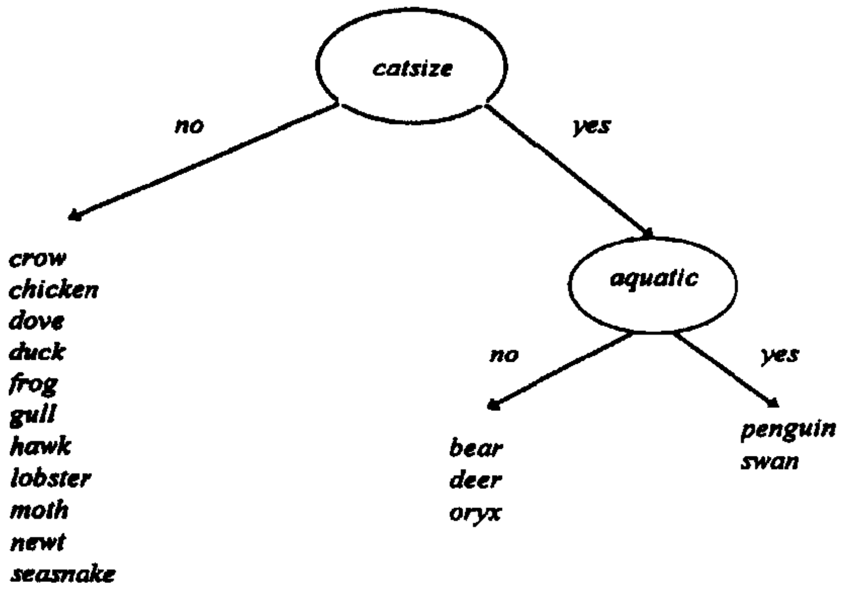

**Fig. 2. The identification tree produced by ID3 to distinguish positive and negative samples once the extra class attribute is added.**

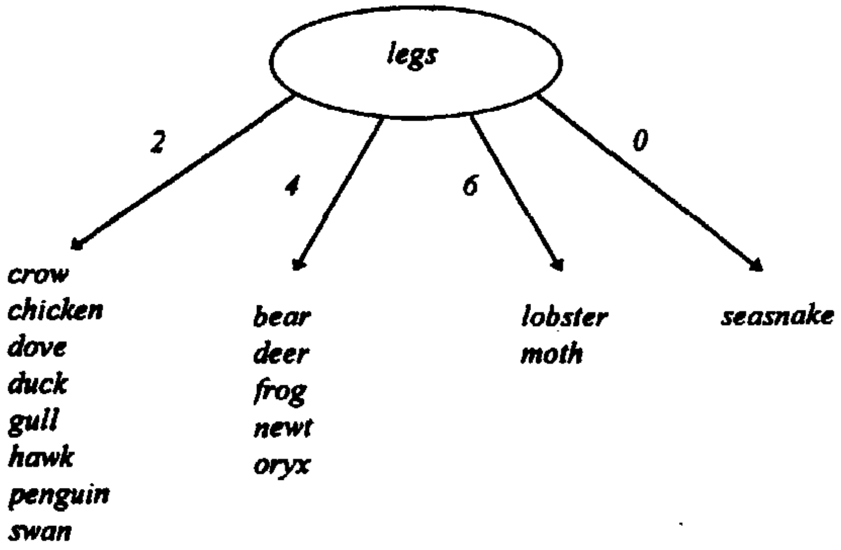

**Fig. 3. The identification tree produced by ID3 to distinguish positive and negative samples for the second example query once the extra class attribute is added.**

**Table 3. The zoo dataset with the new attribute class appended, with values signifying whether the tuple satisfies the second example query**

| Animal | Domestic | Airborne | Aquatic | Legs | Catsize | Class |
|---|---|---|---|---|---|---|
| Bear | no | no | no | 4 | yes | no |
| Crow | no | yes | no | 2 | no | yes |
| Chicken | yes | yes | no | 2 | no | yes |
| Deer | no | no | no | 4 | yes | no |
| Dove | yes | yes | no | 2 | no | yes |
| Duck | no | yes | yes | 2 | no | yes |
| Frog | no | no | yes | 4 | no | no |
| Gull | no | yes | yes | 2 | no | yes |
| Hawk | no | yes | no | 2 | no | yes |
| Lobster | no | no | yes | 6 | no | no |
| Moth | no | yes | no | 6 | no | no |
| Newt | no | no | yes | 4 | no | no |
| Oryx | no | no | no | 4 | yes | no |
| Penguin | no | no | yes | 2 | yes | yes |
| Seasnake | no | no | yes | 0 | no | no |
| Swan | no | yes | yes | 2 | yes | yes |

## 4. Conclusion

What has been described here is a preliminary identification of a novel approach to user guided KDD. It overcomes problems with earlier approaches where different queries which denote the same data samples are given different knowledge expressions. This raised the danger that deep relationships which link the same data samples together are being missed. The approach described here implies that two or more queries, if they return the same data samples, have the same identification tree. Also, the approach described here identifies new concepts not obviously present in the data attributes by allowing the user to name such concepts, knowing that there is a structured way to refer to these concepts through the derived identification trees and associated rules.

More work is required to identify the potential of the approach for extracting conceptual knowledge from the data and its advantages over a data mining approach which attempts to extract knowledge from data independently of any user queries. One possible advantage is that the approach described here does not require the database to contain specific, predefined attributes/classes into which data samples fall. Such attributes/classes are defined by user queries and will change according to queries issued on the database. QGKDD may therefore be of most use when the database consists of purely descriptive information.

## References

[1] Fayyad, U. M. and Uthurusamy, R. "Knowledge Discovery in Databases (KDD-94)." American Association for Artificial Intelligence (AAAI-94). *Workshop Notes* (1994).

[2] Fisher, D. "Knowledge Acquisition via Incremental Conceptual Clustering." *Machine Learning*, 2, No. 2 (1987), 139-172.

[3] Frawley, J., Piatetsky-Shapiro, G. and Matheus, C. J. "Knowledge Discovery in Databases: An Overview." In: *Knowledge Discovery in Databases. AAAI/MIT Press*, (1991), 1-27.

[4] Ioannidis, Y.E., Saulys, T. and Whitsitt, A.J. "Conceptual Learning in Database Design." *ACM Transactions on Information Systems*, 10, No. 3 (1992), 265-293.

[5] Lebowitz, M. "Experiments with Incremental Concept Formation: UNIMEM." *Machine Learning*, 2, No. 2 (1987), 103-138.

[6] Narayanan, A. "Revisable Knowledge Discovery in Databases (RKDD)." *International Journal of Intelligent Systems*. To appear, 1995.

[7] Piatetsky-Shapiro, G. and Frawley, W. J. *Knowledge Discovery in Databases. AAAI Press/MIT Press*. 1991.

[8] Quinlan, J.R. "Induction of Decision Trees." *Machine Learning*, 1, No.1 (1986), 81-106.

[9] Utgoff, P. E. "An Incremental ID3." *Proceedings of the Fifth International Conference on Machine Learning*, 1988.

[10] Winston, P.H. "Artificial Intelligence." 3rd ed. New York : Addison Wesley, 1992.

[11] Wirth, J. and Catlett, J. "Experiments on the Cost and Benefits of Windowing in ID3." *Proceedings of the Fifth International Conference on Machine Learning*, 1988.

[12] Yoon, J. P. and Kerschberg, L. "A Framework for Knowledge Discovery and Evolution in Databases." *Technical Report, George Mason University ISSE*, 1994.

# الاستفسار في توجيه اكتشاف المعرّفة في قواعد البيانات

**سعود سعيد المتحمي**

*قسم علوم الحاسب، كلية علوم الحاسب والمعلومات، جامعة الملك سعود، ص ب ٥١١٧٨،*
*الرياض ١١٥٤٣، المملكة العربية السعودية*

(قدّم للنشر في ١١/١١/١٩٩٥م، وقبل للنشر في ٢٩/٥/١٩٩٦م)

**ملخص البحث** . يتناول هذا البحث إحدى الطرق الحديثة لتقصّي المعرفة من البيانات. حيث تستخدم الإجابات على استفسارات المستفيد لتوجيه اكتشاف تلك المعرفة من البيانات، إذ أن هذه الطريقة الحديثة تختلف عن الطرق السابقة وهي ماتسمى بالطرق الغير توجيهية، والتي تستخدم فقط محتوى وتركيب الاستفسار.

يعرض البحث بعض الأمثلة لتوضيح جدوي استخدام هذه الطريقة والتي أطلق عليها الاستفسار في توجيه اكتشاف المعرفة في قواعد البيانات.