Concept-Based Data Classification in Relational Databases †

Jiawei Han, **Yandong Cai** and **Nick Cercone**
School of Computing Science
Simon Fraser University
Burnaby, British Columbia, Canada
{han, cai, nick}@cs.sfu.ca

Abstract. *Data classification is a process which groups objects with common properties into classes and produces a classification scheme over a set of data objects. Data classification is useful for understanding and organizing database data and building hierarchical schemes in databases. We investigate data classification in relational databases and develop a method for data classification by concept-based generalization. Our method applies an attribute-oriented generalization technique which utilizes the knowledge about data concepts, integrates a data classification process with relational operations, and provides an efficient way for classification of data in relational databases. The characteristics of each class can be extracted automatically in the classification process. Moreover, quantitative information can be registered in the generalization process to assist the classification of data based on database statistics. Our analysis of the classification algorithms shows that the attribute-oriented approach substantially reduces the complexity of data classification in large databases.*

Keywords: *data classification, conceptual clustering, knowledge discovery in databases, attribute-oriented approach*.

1. Introduction

Data classification, a process which groups objects with common properties into classes and produces a classification scheme over a set of data objects, has been studied in cluster analysis, numerical taxonomy and machine learning research [1, 4, 6, 7, 11, 12, 14]. Data classification is also called *data clustering* [6, 14], however, the term *data clustering* has been used in traditional database literature [19] with a different meaning. Data clustering in the database literature refers to the technique which groups data with similar search keys into physically adjacent blocks for efficient access; while that used in concept classification refers to a technique which groups data according to some concept similarity measurement for semantic analysis. In order to construct meaningful classification of data objects, one needs to define a measure of similarity between the objects and then use it to determine classes. Classes are defined as collections of objects whose intraclass similarity is high and interclass similarity is low.

Data classification is fundamental to research in many fields in social and natural sciences. Since databases potentially store a large amount of data, it is important to develop efficient methods for data classification in databases. Although data in a relational database are usually well-formatted and modeled by semantic data models [10], the contents of the data may not be classified. For example, a chemistry database may store a large amount of experimental data in a relational format, but knowledge and efforts are needed to classify the data in order to determine the intrinsic regularity of the data. Clearly, schemas and data formats are not equivalent to conceptual classes. The classification of database data will help us discover data characteristics, summarize data in an understandable manner, and organize data according to

† The work was supported in part by the Natural Sciences and Engineering Research Council of Canada under the operating grants A-3723/A-4309 and a research grant from Centre for Systems Science of Simon Fraser University.

knowledge-oriented structures.

Many traditional data classification methods are based on some numerical taxonomy [1, 18], where clusters are determined solely on the basis of a predefined numerical measure of similarity. To define such a measure, a data analyst determines attributes that are perceived as relevant for characterizing objects under consideration. Vectors of values of these attributes for individual objects serve as descriptions of these objects. Considering attributes as dimensions of a multidimensional description space, each object description corresponds to a point in the space. The similarity between objects can thus be measured as a reciprocal function of the distance between the points in the description space. Such an approach has produced a number of efficient clustering algorithms [1]. However, the approach makes little use of problem domain knowledge, thus it may lack simple conceptual explanations, or even lead to inappropriate classifications [16]. For example, if geographic data are classified according to physical locations only, it may not be beneficial for a land-use analysis, such as the recognition of highways, rivers, airports, etc.

The conceptual clustering method studied in machine learning research [14] classifies objects based on background concepts and clustering goals. In order to cluster objects into conceptual categories, the notion of similarity is replaced by a more general notion of conceptual cohesiveness. The conceptual cohesiveness between two objects depends not only on the attributes of these objects but also on the attributes of nearby objects, and the set of concepts available for describing object configurations. For example, by clustering experimental data based on the knowledge of chemists, interesting relationships among data can be discovered. Conceptual clustering is a powerful approach at clustering data according to the background knowledge. However, the methods developed previously, such as [14], must repeatedly examine data tuple by tuple (a tuple-oriented generalization [3]), which leads to quite inefficient algorithms for use with large databases.

Although the large amount of data stored in relational databases poses challenges for data classification, relational systems provide many attractive features to facilitate various kinds of data operations. In contrast to the data stored in other forms, relational systems store a large amount of information in a structured and organized manner. Each tuple in the database can be viewed as a typed logical formula in a conjunctive normal form. Such uniformity facilitates the development of efficient data classification algorithms. Moreover, relational databases are supported by well-developed implementation techniques, which can be applied to data classification to improve the performance.

In this paper, we explore efficient data classification methods for relational databases. Given a set of data stored in the database, a set of conceptual bias, a preferred number of classes (i.e., a threshold value), and some evaluation criteria, our concept-based classification algorithm constructs a hierarchy of data classes and discovers the conjunctive rules which represent the characteristics of each class. The algorithm applies an attribute-oriented approach which integrates database operations with the classification process and provides a simple, efficient way for concept-based data classification in relational databases.

In Section 2, we present the criteria for data classification in relational databases. In Section 3, we investigate concept-based data classification and develop an algorithm, **DBCLASS**, for the classification of data in relational databases. A discussion of the variations of our method is found in Section 4, a comparison of our method with other data classification techniques is discussed in Section 5, and the concluding remarks are presented in Section 6.

2. Criteria for Data Classification in Relational Databases

A data classification process is strongly influenced by the evaluation criteria for data classification and the background knowledge, such as concept hierarchies, data constraints, etc. A classification task usually consists of the following four primitives: (i) a set of task-relevant data, (ii) a set of *conceptual bias* (concept hierarchy, etc.), (iii) a threshold value, and (iv) a set of criteria for evaluating classification quality.

A preprocessing step should be performed initially to gather data relevant to a specified classification task. Although a large amount of data can be stored in a relational database, only a portion of it is relevant to a specific classification task. For example, to classify commodities in a department store, only the commodities in that store are useful for the classification process. If the task-relevant data are spread over several relations, a preprocessing step, which utilizes selection, projection, join and other relational operations, should be performed to collect the task-relevant data into one data relation.

Next, it is necessary to incorporate the concept hierarchy information for data classification. Each attribute in a data relation may contain a large set of distinct values. Without generalizing data to higher level concepts, the classification results can not be represented in a clear and concise manner. Similar to other data classification tasks [7, 14], it is necessary to provide information about higher level concepts for the data classification process. By organizing different levels of concepts into a taxonomy, a concept space can be partially ordered according to general-to-specific ordering. The most general point in the concept space is the null description, and the most specific points correspond to the specific values of attributes in the database. We restrict our candidate rules to formulas with a particular vocabulary, that is, a basis set which we call the *conceptual bias*, permitting the learned rules to be represented in a simple and explicit form [8].

Usually, the conceptual bias is provided by knowledge engineers or domain-specific experts. In our discussion, we assume that the conceptual bias is given for each attribute, represented as a concept tree, specified using an *is-a* hierarchy †, and stored in a relation table, the *concept hierarchy table*. For example, suppose a *commodity* relation in a *department store* database contains the following attributes: *item#, brand, place_made, category, color*, and *price*. Also, there is a *concept hierarchy table*, as illustrated in Figure 1, which specifies the available concept hierarchies of the attributes, where the notation $\{A_1, ..., A_k\} \subset B$ indicates that B is a generalization of A_i, for $1 \leq i \leq k$, that is, A_i *is_a* B.

It is desirable that classification is represented in a simple and easily understandable form. That is, the number of classes to be formed should be within a small range. Therefore, it is convenient to specify a threshold value, *class threshold*, which represents the maximum number of classes to be classified.

{ 1 .. 99 } ⊂ cheap
{ 100 .. 499 } ⊂ moderate
{ 500 .. 5000 } ⊂ expensive
{ cheap, moderate, expensive} ⊂ Any (Price)
{ red, black, brown, blue } ⊂ dark_color
{ white, yellow, pink, silver } ⊂ light_color
{ light_color, dark_color } ⊂ Any (color)
{ love_seat, sofa_set, ... } ⊂ sofa
{ dining table, kitchen table, ... } ⊂ table
{ queen_size_bed, twin_size_bed, ... } ⊂ bed
{ sofa, table, bed, ... } ⊂ furniture
{ video_recorder, video_camera, ... } ⊂ video_equipment
{ stereo_equipment, AM/FM_radio, ... } ⊂ audio_equipment
{ video_equipment, audio_equipment, ... } ⊂ elec_equip
{ skirt, fur, blouse, ... } ⊂ ladies'_wear
{ tie, sports_coat, ... } ⊂ men's_wear
{ sleeper, pants, ... } ⊂ children's_wear
{ ladies'_wear, men's_wear, children's_wear, ... } ⊂ clothing
{ furniture, elec_equip, clothing, ... } ⊂ Any (Category)
{ Seattle, Chicago, LA, ... } ⊂ USA
{ Tokyo, Osaka, ... } ⊂ Japan
{ Shanghai, Beijing, Tianjin, ... } ⊂ China
{ USA, Japan, China, ... } ⊂ Any (Place)

Figure 1. Concept hierarchy table for a department store database.

† Aggregation and other hierarchies are permissible as well.

Finally, we examine the set of criteria for evaluating classification schemes. The data in each attribute should be generalized in order to obtain concise classification schemes. However, even when this is done, there may still be many possible data classification schemes for a given set of generalized data. Thus some evaluation criteria should be provided to judge the quality of these schemes. Two criteria, *sparseness* and *complexity*, are popularly used in data classification [14]. We examine these criteria through the use of an example.

Category	Place_made	Price	Color
clothing	China	cheap	light_color
clothing	China	moderate	light_color
clothing	China	cheap	dark_color
clothing	China	moderate	dark_color
elec_equip	USA	moderate	dark_color
elec_equip	Japan	moderate	dark_color
elec_equip	USA.	cheap	light_color
elec_equip	Japan	cheap	light_color
elec_equip	Japan	expensive	light_color
elec_equip	China	cheap	light_color
furniture	USA	cheap	dark_color
furniture	USA	expensive	dark_color
furniture	Japan	moderate	dark_color

Table 1. A generalized *commodity* relation.

Suppose Table 1 represents a *generalized relation* of the original data relation *commodity*, which is a relation obtained by applying an attribute-oriented generalization approach [3] to generalize the data in the data relation from a primitive level to a higher-level in the concept hierarchy. Details are discussed in the next section.

In a manner similar to [14], we define an **event** is an object description in the form of a vector of attribute values. The **event space** is the space of all possible such events. An *event space table* is a geometric representation of events and the event space.

Table 1 can be represented geometrically in an event space table, shown in Table 2, in which each slot represents a vector of values with each value deriving from a distinct attribute, and each tuple in the generalized relation is marked by "∗" in the corresponding slot.

A collection of events to be classified into one **(event) class** can be represented as a **(logical) conjunction** of a set of conjuncts. Each **conjunct** is of the form

		USA			Japan			China		
		C	M	E	C	M	E	C	M	E
elec_equip	dark		∗			∗				
	light	∗			∗		∗	∗		
clothing	dark							∗	∗	
	light							∗	∗	
furniture	dark	∗		∗		∗				
	light									

Table 2. The event space table.
(C — cheap, M — moderate, E — expensive)

$$attribute\_name(x) \in \{value\_set\},$$

where *value_set* is a set of values in the attribute *attribute_name*. We define the *entire event space* and the *occupied event space* for a collection of events E in the generalized relation (i.e., an event class) which satisfy a logical conjunction L (i.e, satisfy all the conjuncts in the conjunction) as follows.

Definition. The **entire event space**, P_{entire}, of a logic conjunction L is an event space which represents all the possible tuples in a generalized relation that satisfy the logic conjunction L. It corresponds to the total number of slots in the event space table for the conjunction L. It can be calculated from the logical conjunction by taking the product of V_i, where V_i is the number of distinct values in the i-th attribute of the conjunction ($1 \le i \le n$, where n is the number of attributes in a generalized relation). That is,

$$P_{entire} = \prod_{i=1}^{n} V_i .$$

As an example, we examine the conjunction (*1*).

(*1*) [Price(x) ∈ {cheap, moderate, expensive} /\ Color(x) ∈ {dark_color} /\ Place_made(x) ∈ {USA, Japan} /\ Category(x) ∈ {furniture}].

Since the conjunction contains three values in attribute *Price*(x), one value in *Color*(x), two values in *Place_made*(x) and one value in *Category*(x), the entire event space of the conjunction should be

$$P_{entire} = 3 \times 1 \times 2 \times 1 = 6.$$

In the event space table, Table 2, the conjunction represents the area within the frame whose rows are those covered by *furniture* and *dark*, and whose columns are those covered by *USA* or *Japan*.

Definition. The **occupied event space**, P_{occupy}, of a conjunction L is the space occupied by the events (i.e., the instances occurring in the generalized relation) in the entire event space, P_{entire} of L.

Definition. **Sparseness**, **S**, is the area in the entire event space which is not occupied by the data instances. That is, **S** is the difference between the entire event space and the occupied event space, i.e.,

$$\mathbf{S} = P_{entire} - P_{occupy} .$$

For example, since there are 3 data instances in Table 1 covered by the conjunction (*1*), $P_{occupy} = 3$. Obviously, the *sparseness* $S = 6 - 3 = 3$. In Table 2, within the above outlined frame for P_{entire}, the occupied space is marked by ∗, and the unoccupied space is left blank. The sparseness is the number of blank slots within the frame.

The second criteria that we use in the data classification process is the *complexity*, *C*.

Definition. Let d_i be the number of distinct values in the *i-th* conjunct in a conjunction L, where $1 \le i \le n$, and n is the number of conjuncts in a conjunction. Then the **complexity**, **C**, is the sum of all d_i, i.e.,

$$C = \sum_{i=1}^{n} d_i .$$

In the conjunction (*1*), the number of distinct values in the four conjuncts are {3, 1, 2, 1}. However, since the value set of the attribute *Price* is {*cheap*, *moderate*, *expensive*}, which covers all of the possible (generalized) values for the attribute *Price*, these concepts can be further generalized to *ANY*. Then the values for the attribute *Price* will not play a role in distinguishing classes, thus the attribute can be dropped from the representation of the conjunction. Thus the complexity for the conjunction should be,

$$C = 1 + 2 + 1 = 4.$$

A low sparseness implies that there is a high similarity within each class and a low similarity among different classes. A low complexity indicates that the characteristics of each class are represented by a rule in a simple form. Therefore, the goal of the data classification process is to derive a classification scheme which has a combination of low sparseness and low complexity.

3. Data Classification by Concept-Based Generalization

In this section, we study the data classification process and develop a concept-based data classification algorithm, **DBCLASS**. The algorithm is based on an attribute-oriented generalization technique developed in [3] and a concept-based criteria evaluation technique, influenced by [14]. The method integrates generalization and classification processes and provides a simple, efficient way for concept-based classification of relational data.

We assume that the classification is based on the existing data and knowledge which include the task-relevant data, a conceptual bias, a class threshold (the desired number of classes), and the evaluation criteria. The classification process is divided into two steps: *generalization* and *classification*. First, an attribute-oriented concept tree ascending technique is applied to the task-relevant data relation, which generalizes the concepts of each attribute to a certain level based on the conceptual bias and the class threshold value. Then a classification process is performed on the generalized relation, which creates possible classification schemes, evaluates the schemes based on two criteria, *sparseness* and *complexity*, and selects the appropriate classification scheme.

The process is demonstrated using the following example.

Example 1. Suppose we have a *commodity* relation, Table 3, in a *department store* database which registers the information about imported goods.

A large data relation usually contains a huge set of distinct concepts (attribute values). To obtain a simple and concise classification scheme and derive characteristic rules for each class, we should first generalize the primitive data instances to higher level concepts. This is realized by an attribute-oriented generalization [3] on the task-relevant relation.

The first attribute, "Item#", is the key of the relation. The key value is distinct for each tuple in the relation. If there is no higher level concept provided for such an attribute in the concept tree, the values of the attribute cannot be generalized, and they should be removed in the generalization. Also, other candidate key attributes or nonkey attributes can be eliminated under a similar condition. For example,

Item#	Category	Brand	Place_made	Color	Price
1278	video_camera	Panasonic	Tokyo	silver	755
1717	video_recorder	Sony	Tokyo	black	425
1967	queen_size_bed	Omaha	Chicago	black	799
2164	AM/FM_radio	Sanyo	Tokyo	yellow	88
2252	twin_size_bed	Comfort	Chicago	black	95
3353	sleeper	Star	Shanghai	yellow	8
3507	kitchen_table	Woodman	Seattle	brown	78
4456	love_seat	Lazy_boy	Osaka	brown	407
4528	AM/FM_radio	Sunset	LA	silver	93
5466	blouse	Happy	Tianjin	white	125
6392	AM/FM_radio	Panda	Beijing	white	62
7373	video_recorder	RCA	Chicago	black	466
7588	sports_coat	Fortune	Shanghai	Blue	156
8002	skirt	Mini	Beijing	red	45
8344	video_camera	RCA	Seattle	brown	455
9005	stereo_equipment	Sony	Osaka	Brown	388
9345	fur	Firebird	Tianjin	white	325

Table 3. The *commodity* relation in the *department store* database.

the attribute *Brand* should be eliminated in the generalization since there is no higher level concept provided on this attribute.

We then examine the remaining four attributes. Although the four attributes contain many distinct values, the values are associated with some higher level concepts in the concept hierarchy table of Figure 1. By examining the values attribute-by-attribute and substituting the values in each attribute by their higher level concepts in the corresponding concept tree, e.g., from "skirt" to "ladies'_wear" and from "black" to "dark", the relation is generalized. In general, we have

Strategy 1. (**Attribute-oriented generalization**) *The generalization is performed on each attribute by removing the attribute if there is no higher-level concept available for the attribute, or by ascending the concept tree (i.e., substituting the value of the attribute in each tuple by its corresponding higher level concept) otherwise*.

Rationale:

Attribute removal corresponds to the generalization rule, *dropping conditions* [13]. Consider a tuple as a set of conjuncts in the logical forms; an attribute value together with its attribute name form one of the conjuncts. By removing a conjunct, a constraint is eliminated and the concept is generalized. If there are a large set of distinct values for an attribute, the large set of values must be generalized. However, if there is no higher level concept provided for the attribute, it cannot be further generalized by ascending the concept tree. Therefore, the attribute should be eliminated in generalization. Attribute removal can also be viewed as a generalization of all attribute values to the most general concept ANY and then removed from the representation.

Concept tree ascension corresponds to the generalization rule, *climbing generalization trees* [13]. If there exists a higher level concept for the value in the concept tree, the substitution of the value in each tuple in the relation by the corresponding higher level concept makes the tuple cover more cases than the original one, and thus it generalizes the tuple. □

Notice that it is often necessary for some attribute values to be generalized by climbing up a concept tree several times in order to derive a simple classification scheme. To control the generalization process and integrate it with the classification process, it is necessary to provide a **class threshold**, which is the upper bound on the number of classes to be partitioned. Then the generalization can be repeatedly performed on an attribute (*without examining other attributes*) until the number of distinct values in the attribute is under the specified threshold value. For instance, if the threshold value is set to three in this example, the concepts in the attribute *Place_made* should be generalized by two levels. In general, we have

Strategy 2. (**Threshold control**) *If the number of distinct values of an attribute is larger than the specified class threshold value, further generalization on this attribute should be performed*.

Rationale:

According to the definition of the class threshold, the threshold value, T, is the upper bound of the number of classes to be partitioned. When the number of distinct values in an attribute of a relation is larger than T, the resulting classification scheme contains more classes than T. Thus further generalization should be performed on this attribute. □

Since different attribute values can be substituted by the same higher level concepts in the generalization process, a set of tuples may be generalized to the same generalized tuple. By eliminating such redundant tuples in the generalized relation, the size of the generalized relation is reduced.

A detailed discussion of the attribute-oriented generalization can be found in [3]. For the *commodity* relation Table 3, if the threshold value is set to 3, the attribute-oriented generalization will derive the generalized relation, Table 1 (with redundant tuples eliminated).

After the generalization process, classification is performed on the generalized relation. The possible classification schemes can be generated and evaluated using an attribute-oriented approach. Each classification scheme is generated based on the (generalized) values of one attribute, called the **classifying attribute**.

Taking *Category* as the classifying attribute, we examine the classification process.

First, the tuples in the generalized relation are clustered based on the values in the classifying attribute. That is, based on the three generalized values of the attribute *Category*, the data instances in Table 1 are grouped into three classes: *furniture class*, *clothing class* and *elec_equip class*. Then a conjunctive rule for each class is derived as follows:

(1) If the values of an attribute in a class cover the entire set of the concept in the corresponding level of the concept tree, this attribute should not be included in the conjunctive rule. For example, in the *elec_equip class*, the colors of the items vary from *dark* to *light*. Since *dark* and *light* cover all of the possible colors in the concept tree, the attribute *Color* should not appear in the conjunctive rule for the *elec_equip Class*.

(2) Otherwise (i.e., the values do not cover the entire concept set), the attribute should be included in the conjunctive rule, and all of the (generalized) values in the attribute should be included in the attribute value set. For example, in the *furniture class*, the items are either from *USA* or from *Japan*, but not from *China*. To represent the characteristic of this class, "Place_made $\in$ {USA, Japan)" should appear in the conjunctive rule for the *furniture class*.

In general, we have

Strategy 3. (**Formation of conjunctive rules**) *If the (generalized) data values appearing in an attribute cover all of the possible values of the attribute in the corresponding generalization hierarchy, the attribute should be dropped from the representation of the conjunctive rule; otherwise, the attribute and the set of associated (generalized) values should be included in the conjunctive rule of the class*.

Rationale:

If the set of generalized data values appearing in an attribute covers the entire concept set for the attribute, the value in this attribute could be "ANY" value of the set, and thus it is useless to be included in the conjunctive rule. On the other hand, if the values appearing in the attribute do not cover the entire concept set, it indicates that there exist only some specific values for the attribute in this class. By including these specific values in the conjunctive rule, the classification is precisely characterized and the boundary of the class is clear. Thus the attribute and its associate value set should be included in the conjunctive rule. □

There are four attributes in the generalized relation, Table 1. By performing the scheme generation process on four different classifying attributes, four classification schemes are derived as shown in Table 4.

According to the algorithm, there are n candidate classification schemes for a generalized relation with n attributes. However, the quality of the classification schemes may differ greatly. To select a high quality classification scheme, two criteria, *sparseness* and *complexity*, can be used to examine and compare the candidate schemes.

Taking *Scheme 1* as an example, we calculate its sparseness and complexity. First, the entire event space for Scheme 1 is the sum of the entire event space of its component classes. For the furniture class, P_{entire} = 6 (see Section 2). Similarly, for the clothing class, P_{entire} = 1 (for "Category") $\times$ 2 (for "Color") $\times$ 1 (for "Place_made") $\times$ 2 (for "Price") = 4, and for the *elec_equip* class P_{entire} = 1 (for "Category") $\times$ 2 (for "Color") $\times$ 3 (for "Place_made") $\times$ 3 (for "Price") = 18. Thus the total P_{entire} is,

$$P_{entire} = 6 + 4 + 18 = 28.$$

The sparseness of a scheme, $S = P_{entire} - P_{occupy}$. For Scheme 1, its occupied space is (which can be easily calculated from Table 2),

$$P_{occupy} = 6 + 4 + 3 = 13.$$

Therefore, its sparseness is

$$S = 28 - 13 = 15.$$

Scheme 1 [Classifying attribute = "Category"]. It contains 3 classes:

1 : [Category(x) $\in$ {elec_equip}]

2 : [Category(x) $\in$ {clothing} /\ Place_made(x) $\in$ {China} /\ Price (x) $\in$ {cheap, moderate}]

3 : [Category(x) $\in$ {furniture} /\ Color(x) $\in$ {dark_color} /\ Place_made(x) $\in$ {USA, Japan}]

Scheme 2 [Classifying attribute = "Place_made"]. It contains 3 classes:

1 : [Place_made(x) $\in$ {USA} /\ Category(x) $\in$ {elec_equip, furniture}].

2 : [Place_made(x) $\in$ {Japan} /\ Category(x) $\in$ {elec_equip, furniture}].

3 : [Place_made(x) $\in$ {China} /\ Category(x) $\in$ {elec_equip, clothing} /\ Price(x) $\in$ {cheap, moderate}].

Scheme 3 [Classifying attribute = "Price"]. It contains 3 classes:

1 : [Price(x) $\in$ {cheap}].

2 : [Price(x) $\in$ {moderate}].

3 : [Price(x) $\in$ {expensive} /\ Category(x) $\in$ {elec_equip, furniture} /\ Place_made(x) $\in$ {USA, Japan}].

Scheme 4 [Classifying attribute = "Color"]. It contains 2 classes:

1 : [Color(x) $\in$ {dark}].

2 : [Color(x) $\in$ {light} /\ Category(x) $\in$ {elec_equip, clothing}].

Table 4. The candidate classification schemes generated from the generalized relation (Table 1).

The complexity of a scheme is the sum of the complexity of its component classes. The complexity of each class is calculated by summing up the number of distinct values in all the conjuncts in the conjunction. Obviously, the complexity for Scheme 1 should be

$$C = 1 + 4 + 4 = 9.$$

The *sparseness* and *complexity* for other schemes can be computed based on the same calculation method. Then the next step is to select a desired classification scheme from these candidates. Since a low sparseness implies that there is a high similarity within each class and a low similarity among different classes. A low complexity indicates that the characteristics of each class are represented by a rule in a simple form. Therefore, if a classification scheme has both a low sparseness and a low complexity, it should be selected as the desired classification scheme.

In general, the desired classification scheme should be the one with the minimum combined value of the sparseness and complexity among all the candidate schemes. If $combined(sparseness, complexity)$ is the function which derives such a combined value, the desired scheme should be

$$\min(\, combined(sparseness, complexity)\,).$$

Notice that *combined* can be sum, product, weighted sum, or other functions, which is to be discussed in Section 4.1. Let $combined(sparseness, complexity)$ be the sum of the two component values. The sparseness, complexity and their combination values for the four classification schemes is presented in Table 5.

Scheme#	*Sparseness*	*Complexity*	*combined* (*Sparseness* , *Complexity*)
Scheme 1	15	9	24
Scheme 2	19	11	30
Scheme 3	31	7	38
Scheme 4	32	4	36

Table 5. The sparseness, complexity and their combination for the four classification schemes.

Since Scheme 1 has the lowest value for the combination of sparseness and complexity measurements, it should be the best classification scheme for the given data. The classification and its coverage can be represented geometrically by outlining the class boundary frame (using doubly lined frame) for each class in Table 2, which is demonstrated in Table 6.

		USA			Japan			China		
		C	M	E	C	M	E	C	M	E
elec_equip	dark		*			*				
	light	*			*		*	*		
clothing	dark							*	*	
	light							*	*	
furniture	dark	*		*		*				
	light									

Table 6. The event space table.
(C — cheap, M — moderate, E — expensive)

Therefore, we have the following strategy for the selection of a classification scheme.

Strategy 4. (**Selection of a classification scheme**) *The classification scheme which has a combination of low sparseness and low complexity should be selected in the data classification.*

Rationale:

Based on the reasoning in the above analysis. □

Notice that each conjunction generated in the classification process represents some general characteristics of the corresponding class. For example, the conjunction for the third class of Scheme 1 ($Category(x) \in$ {furniture}) is,

[Category(x) ∈ {furniture} /\ Color(x) ∈ {dark_color} /\ Place_made(x) ∈ {USA, Japan}].

It implies the following characteristic rule: if x's category is furniture, it must be a dark_color and be made either in USA or in Japan, that is,

$$\forall (x) \ \ Category(x) \in \{furniture\} \rightarrow$$
$$Color(x) \in \{dark\_color\} \wedge Place\_made(x) \in \{USA, Japan\}.$$

In summary, we present the data classification algorithm below.

Algorithm 1. *DBCLASS — A DB classifier which classifies data in relational databases by concept-based generalization.*

Input. (i) A set of task-relevant data stored in a relation table,
(ii) a concept hierarchy table,
(iii) the evaluation criteria, and
(iv) the threshold value (T).

Output. A classification scheme with a desired combination of low sparseness and low complexity.

Method.

Step 1. *Attribute-oriented induction*.

/* Suppose the relation, P, consists of a set of attributes, A_i, where $1 \le i \le n$, and n is the number of attributes in relation P. T is the threshold value, and d_i is the number of distinct values in attribute A_i. */

begin
 for each attribute A_i $(1 \le i \le n)$ **do**
  while $d_i > T$ **do**
  begin
   if there is no higher level concept of A_i
    then remove attribute A_i
    else substitute the values by its corresponding minimal generalized concept;
   eliminate redundant tuples;
  end
end. {*Attribute-oriented induction*}

Step 2. *Scheme generation*.

/* Suppose the generalized relation, P, consists of a set of attributes, A_i, where $1 \le i \le n$, and n is the number of attributes in the generalized relation P. */

begin
 for each attribute A_i $(1 \le i \le n)$ **do** /* A_i is the classifying attribute */
 begin
  cluster the tuples in P into j classes based on the values of attribute A_i;
  /* Note: j is the number of distinct values of Attribute A_i in P and $1 \le j \le T$. */
  for each class C_j **do**
   form a conjunctive rule for the class based on the tuples in the class;
   /* Note: This is performed as follows:
    if the distinct data value set of an attribute does not cover the whole set of values in the class
    then include the attribute and its associated value set as a conjunct in the conjunctive rule. */
 end
end.

Step 3. *Examine and select classification schemes*.
begin
 Calculate the sparseness and the complexity for each scheme;
 Select the scheme with the smallest combined value of sparseness and complexity.
end.

Theorem 1. *Algorithm DBCLASS derives the desired classification scheme*.

Proof sketch.

Step 1 generalizes the original data relation by adopting Strategy 1 (*attribute-oriented generalization*) and Strategy 2 (*threshold control*). Step 2 uses the generalized relation and generates the candidate classification schemes based on Strategy 3 (*conjunctive rule formation*). Step 3 evaluates the candidate classification schemes and selects the desired one based on the definitions of the evaluation criteria and Strategy 4 (*selection of a classification scheme*). The rationales of these strategies have been presented in this section. Thus we have the theorem. □

4. Variations of the Concept-Based Data Classification Algorithm

The DBCLASS algorithm discussed above is a primitive concept-based database data classification algorithm. It can be extended in many ways based on user requirements, threshold preferences, classification criteria, conceptual hierarchies, and data distribution. In this section, we study two issues on the extensions and variations of the algorithm: (i) the variations of the classification primitives; and (ii) classification based on quantitative information associated generalization.

4.1. Variations of the classification primitives

The DBCLASS algorithm performs data classification based on four primitives: a set of task-relevant data, a set of conceptual bias, a threshold value, and a set of classification criteria. Here we examine the variations of the algorithm based on different conceptual biases, threshold values, and classification criteria, and leave the impact of different data distribution to the next subsection.

Algorithm DBCLASS assumes that a concept hierarchy is given for each attribute to be generalized. This is necessary for nonnumerical attributes since there are usually a large set of distinct values for each attribute in a data relation, and it is impossible to construct a concise and simple classification without generalizing low level concepts to higher level ones, which requires the information about generalization hierarchies. However, it may not be mandatory to provide generalization hierarchies for numerical attributes since appropriate intervals can be automatically constructed based on the similarity measurement of the numerical values. There have been many well-developed similarity-based numerical data clustering analysis algorithms which can be used for this purpose [1, 18].

Algorithm DBCLASS requires that the number of classes, i.e., the threshold value, is specified in advance. Since the best number of classes to be formed is usually unknown, it is useful to try several thresholds in the classification process in order to obtain a desired concept classification [2, 15, 16].

For most purposes, it is desired that the classification formed be simple and easy to understand. With this in mind, the number of classes that stem from any node of the classification hierarchy can be assumed to be in modest range such as from 2 to 8. With this small range, it is computationally feasible to repeat the whole classification process for every number in the range. The solution that optimizes the score on the classification quality criterion indicates the best number of classes to be formed. That is, the DBCLASS algorithm can be applied repeatedly to one classification task with the number of classes (i.e., the threshold value) varying from 2 to 8. Then we compare the quality of different classification schemes and select the one with the best quality of classification.

Moreover, the algorithm classifies concepts at only one level. To obtain a hierarchy of classes, the algorithm can be generalized by repeatedly applying the algorithm from the top level down, that is, first classifying the database data at the top level and then performing the classification for each newly formed subclass. By doing so, database data can be organized into a hierarchy by conceptual clustering analysis.

Another interesting variation is to introduce negative operators for classification rules [11, 14]. It reduces complexity of rules in some cases. However, there are usually no negative data stored explicitly in databases. It is an interesting research issue to incorporate negative operators in the classification of database data. A preliminary study of this issue is in [17].

Finally, we examine the criteria for the evaluation of classification schemes. In our example, the algorithm treats sparseness and complexity equally in the calculation of their combined values. A user may wish to use other functions in computation or give more preference to sparseness than complexity or vice versa. Moreover, since a user usually knows better about the task and may prefer to emphasize some attributes and de-emphasize others in the classification process. Therefore, the classification evaluation criterion of the DBCLASS algorithm can be modified accordingly. For example, we may weigh some attributes more than others and adjust weights dynamically or interactively in the classification process.

4.2. Classification based on quantitative information-associated generalization

The large amount of data stored in a database may have diverse distributions. Generalization on such kind of data may result in a wide distribution of the generalized data which may occupy most of the event space. It is difficult to achieve high classification quality by direct application of the DBCLASS algorithm on such data distributions.

However, it is often the case that the majority of generalized data may be clustered in some portion of the event space while the other portions of the event space may contain only scattered data. To avoid that a small amount of scattered data distorts the whole picture and hence the classification process, the data which appear occasionally in some scattered portions should be treated as exceptional data or noise in the classification process. Such exceptional data can be handled by the incorporation of quantitative information in the generalization and classification processes.

The DBCLASS algorithm can be modified to incorporate quantitative information in the analysis. Similar to the discovery of quantitative rules from large databases [9], an additional attribute, *vote*, can be associated with each generalized tuple to indicate the number of tuples in the initial data relation generalized to the current one in the generalized relation.

Definitions. The **vote** of each tuple in the initial data relation is assumed to be one. The **vote** of a generalized tuple is the sum of the votes of the lower level tuples which are generalized to the current one. Notice that two tuples are *identical* if they have the same corresponding attribute values without considering their *vote*s.

Obviously, we have

Strategy 5. (**Vote propagation**) *The value of the vote of a tuple is carried to its generalized tuple and the votes are accumulated in the preserved tuple when other identical (thus redundant) tuples are removed in generalization.*

Rationale:

Based on the definition of *vote*, the vote of each generalized tuple must register the number of the tuples in the initial data relation generalized to the current tuple. To keep the correct number of votes registered, the *vote* value in each tuple should be carried in generalization. Moreover, the *vote* of a redundant tuple should be added to that of the preserved tuple when the redundant tuple is removed from the generalized relation. □

Data in a database may be distributed along the full spectrum of the possible values. Without using quantitative information, it is impossible to obtain a meaningful classification for such kind of data. However, using the quantitative information, various kinds of techniques can be developed to extract meaningful classification schemes. A simple method is introduced here.

The method treats the data which occur rarely in some event space as exceptional or noise data and filters them out before the classification process. This is accomplished by calculating the *frequency ratio* for each event space and filtering out the *exceptional* data using a *noise filter threshold*.

Definitions. The **frequency ratio** is the ratio of event occurrence frequency versus the total number of tuples in the database, that is,

$$r_{frequency} = event\_occurrence\_frequency \ / \ number\_of\_events\_in\_DB.$$

The **noise filter threshold** is a small percentage number which is used to filter out those events in the event space table with a very low frequency ratio (i.e., below the specified noise filter threshold).

Example 2. Suppose there are 3000 tuples in the *commodity* relation of Example 1, and the attribute-oriented generalization produces Table 7 in which quantitative information is associated by *votes*.

		USA			Japan			China		
		C	M	E	C	M	E	C	M	E
elec_equip	dark	9	264	3	6	192	5	12	8	10
	light	290		4	367		52	185		9
clothing	dark		11	7	5	10		85	228	1
	light	7	2		3	8		64	345	7
furniture	dark	153		242	13	321	7	12	5	3
	light		12	1		6	4	15	6	2

Table 7. The event space table with quantitative information associated.

We calculate the frequency ratio of each entry in the event space table, Table 7, which results in Table 8.

Suppose the *noise filter threshold* is set to 1%. The noise filtering operation is performed as follows. If the frequency ratio of an event slot is less than 1%, the slot is cleared; otherwise, it is marked by "∗". By performing the noise filtering operation on the event space table Table 8, the same space event table Table 2 is obtained, which represents an event space table containing only nonexceptional events.

After this, the process of generation and evaluation of classification schemes should be the same as the one discussed in Example 1, which results in the same classification scheme. □

5. A Comparison with Other Data Classification Methods

The DBCLASS algorithm and its variations provide an efficient approach for the classification of relational database data by concept-based generalization. We analyze the computational complexity of the method and compare it with other data classification methods.

As shown in the DBCLASS algorithm, the classification process can be divided into two steps: (i) attribute-oriented generalization, and (ii) generalized data classification. The total cost of computation should be the sum of the computational cost at each step.

Let N be the number of tuples in the task-relevant relation, A be the number of attributes in it, and H be the maximum height of the concept trees. The time complexity of the DBCLASS algorithm is analyzed as follows.

At Step 1 (attribute-oriented generalization), since the concepts in each attribute can be generalized at most H times, the cost of the generalization process for A attributes is $A \times H \times N$. After the concepts of all the attributes are generalized to the threshold level, the algorithm examines and removes the redundant tuples, which costs $O(N \log N)$ in the worst case.

Suppose there are only N' tuples left in the generalized relation. At Step 2 (scheme formation), it takes $O(N' \log N')$ time (in the worst case) to sort and group the classification schemes and $O(A \times N')$ time to examine each value of each attribute to form a conjunctive rule for each classification scheme. Since there are A possible classification schemes for A classifying attributes, the total cost of the classification process will be $A \times O(N' \log N') + A \times O(A \times N')$.

Notice that A and H are usually much smaller than N or N'. Therefore, the worst-case time complexity for the generalization process is $O(N \log N)$ and that for the classification process is $A \times O(N' \log N')$, where N and N' are the total number of tuples in the original relation and the generalized relation respectively.

Then we examine other data classification methods. Traditional numerical data clustering algorithms cannot classify data based on concepts or knowledge [1, 14]. Most conceptual classification algorithms in the literature [7, 14, 16] are tuple-oriented algorithms. A tuple-oriented algorithm examines data in the database tuple by tuple and performs generalization and classification based on the comparison of tuple values with the intermediate generalization results. Since the number of the possible tuple combinations is exponential to the number of tuples in the relevant data sets, the worst case complexity of

		USA			Japan			China		
		C	M	E	C	M	E	C	M	E
elec_equip	dark	0.3%	8.8%	0.1%	0.2%	6.4%	0.17%	0.4%	0.27%	0.33%
	light	9.67%		0.13%	12.23%		1.73%	6.17%		0.3%
clothing	dark		0.37%	0.23%	0.17%	0.33%		2.83%	7.6%	0.03%
	light	0.23%	0.07%		0.1%	0.26%		2.13%	11.5%	0.23%
furniture	dark	5.1%		8.07%	0.43%	10.7%	0.23%	0.4%	0.17%	0.1%
	light		0.4%	0.03%		0.2%	0.13%	0.5%	0.2%	0.07%

Table 8. The event space table with frequency ratio in each slot.

the generalization and classification process is exponential to the size of the relevant data sets [3].

Moreover, our approach integrates the generalization and classification processes with database operations. Most of the operations in the algorithms are traditional relational database operations, such as selection, join and projection (for extracting relevant data and removing attributes), tuple substitution (for ascending concept trees), and intersection (for discovering overlapping tuples). Such operations are set-oriented and have been efficiently implemented in relational systems. While most other classification algorithms suffer from inefficiency problems in a large database environment due to their tuple-oriented exhaustive search, our approach provides an efficient solution to data classification in databases [5].

Furthermore, our algorithms may use quantitative information to perform data classification when exceptional or noise data are present, which is frequently the case in large databases. Therefore, our method provides additional flexibility over the traditional conceptual clustering analysis methods.

6. Conclusions

Data classification is a useful technique for understanding and organizing database data and building hierarchical schemes in databases. In this paper we developed a method for the classification of data in relational databases by concept-based generalization. The method performs an attribute-oriented generalization on the data relation and an attribute-oriented conceptual classification on the generalized data. The classification process groups similar generalized concepts into classes and produces a classification scheme with a high classification quality among the candidate classification schemes. The technique utilizes knowledge about data concepts, integrates a data classification process with a generalization process, and provides an efficient way of classification of data in relational databases. The characteristic rule of each class can be extracted automatically in the classification process. Moreover, quantitative information can be registered in the generalization process to assist the classification of data based on database statistics. Our analysis of the algorithms demonstrates that attribute-oriented approach substantially reduces the complexity of data classification in large databases.

Based on our previous implementation of the DBLEARN system [3] which performs attribute-oriented generalization to extract knowledge rules in relational databases, we are currently working on the implementation and testing of the DBCLASS algorithm for the large database environment. There are many issues need to be further researched. For example, our classification schemes does not allow conjunctive attributes as classifiers. Therefore it cannot handle nonlinear classification. Other issues such as the integration of numerical clustering techniques with the DBCLASS algorithm in the presence of numerical data attributes, the application of negative operators in the classification process, other classification evaluation techniques using associated quantitative information, automatic generation of concept hierarchies, etc. are interesting topics for the future research. We are investigating into these issues and hope to report our progress in the future.

References References

1. M. R. Anderberg, *Cluster Analysis for Applications*, Academic Press, 1973.
2. L. B. Booker, D. E. Goldberg and J. H. Holland, Classifier Systems and Genetic Algorithms, *Artificial Intelligence*, **40**, 1989, 235−282.
3. Y. Cai, N. Cercone and J. Han, Attribute-Oriented Induction in Relational Databases, *in G. Piatetsky-Shapiro and W. J. Frawley (eds.), Knowledge Discovery in Databases, AAAI/MIT Press , 1991, 213−228.*
4. Y. Cheng and K. S. Fu, Conceptual Clustering in Knowledge Organization, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **7**(5), September 1985, 592−598.
5. T. G. Dietterich and R. S. Michalski, A Comparative Review of Selected Methods for Learning from Examples, *in Michalski et. al. (eds.), Machine Learning: An Artificial Intelligence Approach, Vol. 1, Morgan Kaufmann, 1983, 41−82.*
6. D. Fisher and P. Langley, Approaches to Conceptual Clustering, *Proc. 9th Int. Joint Conf. AI*, Los Angeles, CA, August 1985, 691−697.
7. D. Fisher, Improving Inference Through Conceptual Clustering, *Proc. 1987 AAAI Conf.*, Seattle, Washington, July 1987, 461−465.
8. M. Genesereth and N. Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann, 1987.
9. J. Han, Y. Cai and N. Cercone, Discovery of Quantitative Rules from Large Databases, in Z.W. Ras, M. Zemankova and M.L. Emrich (eds.), *Methodologies for Intelligent Systems, 5, North-Holland, 1990, 157−165.*
10. R. Hull and R. King, Semantic Database Modeling: Survey, Applications, and Research Issues, *ACM Comput. Surv.*, **19**(3), 1987, 201−260.
11. P. Langley and S. Sage, Conceptual Clustering as Discrimination Learning, *Proc. 5th Canadian Conf. Aritificial Intelligence*, London, Ontario, 1984, 95−98.
12. P. Langley, J. Zytkow, H. Simon and G. Bradshaw, The Search for Regularity: Four Aspects of Scientific Discovery, *in Michalski et. al. (eds.), Machine Learning: An Artificial Intelligence Approach, Vol. 2, Morgan Kaufmann, 1986, 425−469.*
13. R. S. Michalski, A Theory and Methodology of Inductive Learning, *in Michalski et. al. (eds.), Machine Learning: An Artificial Intelligence Approach, Vol. 1, Morgan Kaufmann, 1983, 83−134.*
14. R. S. Michalski and R. Stepp, Automated Construction of Classifications: Conceptual Clustering Versus Numerical Taxonomy, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **5**, 1983, 396−410.
15. G. Piatetsky-Shapiro, Discovery, Analysis, and Presentation of Strong Rules, in G. Piatetsky-Shapiro and W. J. Frawley (eds.), *Knowledge Discovery in Databases, AAAI/MIT Press, 1991, 229−238.*
16. S. C. Shapiro, *Encyclopedia of Artificial Intelligence*, John Wiley and Sons, Inc., 1990.
17. C. Shum and R. Muntz, Implicit Representation for Extensional Answers, *Proc. 2nd Int. Conf. Expert Database Systems*, Vienna, VA, April 1988, 497−522.
18. R. R. Sokal and R. H. Sneath, *Principles of Numerical Taxonomy*, W. H. Freeman, 1963.
19. J. D. Ullman, *Principles of Database and Knowledge-Base Systems, Vols. 1 2*, Computer Science Press, 1989.