CHAPTER 4

METHODOLOGY AND IMPLEMENTATION OF THE AUTOMATED DATA MINING SYSTEM

This chapter explains the various methodologies implemented in this research work and the different algorithms used in the design and development of an automated data mining system for object oriented data. This chapter starts with the methodology of the Automated Data Mining System used in this research work. It also focuses on the different algorithms like vertical partitioning algorithm, ranking algorithm and the ranking factors used by the ranking agent to rank the attributes used in this research work. Various clustering algorithms for numerical attributes, categorical attributes and for object oriented data are discussed. This chapter focuses on how data mining system and agents are implemented using Java, Jappl and Jat.ite to implement the user interface agent, partitioning agent, ranking agent and data mining agent. Data base used in this research work is also presented at the end of this chapter.

4.1 Methodology of the Automated Data Mining System

In order to develop the automated data mining system the following are the methodologies considered in this research work. The methodology of the Automated Data Mining System for an Object Oriented Data consists of four steps. They are:

a) Identifying the properties of Object Oriented Data

b) Vertical partitioning of Object Oriented Data
c) Ranking the attributes

d) Selection of clustering algorithm and clustering the object oriented data

An Object Oriented data set of the Karunya University, Campus Management system is considered for this research work. The above said methodologies are used in the data set for this research work.

a) Identifying the properties of Object Oriented Data

As soon as the user submits the query or gives the input or high level goal or objectives or user’s hint, the user interface agent present in the data mining system senses the environment and also analyzes the Object Oriented data set. When such a goal or event arises, the user interface agent determines what course of action to take. If the agent already believes that the goal or event has been handled, it looks through its plans to find whether the event has already taken place. In this case the user interface agent will see whether the closely related attributes are identified to one partition. User interface agent will also check the object oriented data has been already partitioned or not. The user interface agent will see its beliefs about the relationships between the objects, classes, attributes and the importance of each of the attributes. If happened already, the previous results were prompted to the user. If, it did not happen already it will execute the plan. The plan is, the instructions the user interface agent follows is to achieve its goals based on the user query.

Based on the submission of the query by the user the user interface agent classifies them according to the following classification. The user interface agent
identifies the relationships between the objects, classes, attributes and the importance of each of the attributes. The user interface agent also identifies the type of the attributes. After the user interface agent analyzes and classifies the object oriented database based on the user’s high level goal or objectives or user’s hint the action takes place. The user interface agent communicates the analyzed and classified information to the partitioning agent.

b) Vertical partitioning of Object Oriented Data

As the second step the vertical partitioning is carried out on the Object Oriented Data. The partitioning agent present in the automated data mining system helps in partitioning the data. An Object Oriented data with simple class consisting of simple attributes is taken for vertical partition. With the simple class, vertical partitioning aims in splitting a class such that all attributes of a class that are closely related to each other are grouped together.

Partitioning agent and the user interface agent communicates with each other. The user interface agent communicates the analyzed and classified information to the partitioning agent. Then, with the help of partitioning processor, the partitioning agent partitions the object oriented data. The partitioning agent will decide the number of partitions based on the relationship between the attributes, classes, objects, and methods analyzed and classified by the user interface agent. Once the Object Oriented data is partitioned, the ranking algorithm is applied over the attributes, so that the highest ranked attributes are taken for clustering.
c) Ranking the attributes

The object oriented data is partitioned by the partitioning agent. The partitioning is carried out based on the attributes that are closely related to each other. After the partition is carried out, ranking of attributes are essential in order to choose the correct attributes. Ranking agent helps in ranking the attributes. Ranking is performed based on the Query weight and a Scoring function.

When ranking agent is faced with such a goal or query it determines what course of ranking action to perform. If the ranking agent already believes that the goal or event has been handled already (as may happen when it is asked to do something that it believes has already been achieved) retrieves the same ranking method that has been used. It also looks through its other ranking factors to find those that are relevant to the request and applicable to the situation. The highest ranked attributes are taken for clustering.

d) Selection of Clustering algorithm and clustering the object oriented data

Data Mining Agent used in the automated data mining system has specific data mining methods and algorithms. This agent activates and manages data mining algorithms such as cluster analysis algorithms. The attributes that are ranked by the ranking agent are given to a data mining agent. The data mining agent used in the automated data mining system will automatically choose the appropriate clustering algorithm for the ranked data. A wide variety of clustering
algorithms are used in this research work, which are listed in the following sections:

4.2 Vertical Partitioning Algorithm

Vertical partitioning is an important technique in which attributes of a relation assigned to partitions, is aimed at improving database performance. The research work concentrates on non-overlapping vertical partitioning, where there is no overlap in the attributes of the vertical partitions. Vertical partitioning, partitions a class such that all attributes and methods of the class, which are closely related to each other. Partitioning agent used in this research work identifies the closely related attributes. A vertical partitioning algorithm is used in this research work which is a procedure for obtaining a partition set.

A vertical partitioning of a class C in an object-oriented database defines both structural and behavioral properties. The structural properties are represented by a set of instance variables I = {i_1, i_2, ..., i_n} results in a set of vertical class partitioning V = {V_1, V_2, V_3, ..., V_n} while behavioral properties are embodied by a set of methods M = {m_1, m_2, ..., m_n}; the latter are used to access and manipulate objects in class C's warehouse. For the former, each instance variable of an object is instantiated by using a value from its domain class. Vertical class partition V_j has a non empty subset of instance variables i^{(j)} = {i^{(j)}_1, i^{(j)}_2, ..., i^{(j)}_n_j} and each i^{(j)}_q ∈ I, q=1, ..., n_j where n_j is the number of instance variables in the jth vertical partition.
4.2.1 Internal Representation of Vertical Fragments

A representation of vertical partitioning has been reported in this research work using a student data from the Campus Management System of Karunya University with the following attributes. A partition set consists of subsets of attributes and methods which represent the vertical partitioning of a given class.

```java
Class Student
{ Registernumber Char [10];
  Initial Char [5];
  FirstName char [20];
  MiddleName char[20];
  LastName char[20];
  Dob char[20];
  Gender char[5];
  Country char [20];
  Religion char[20];
  Denomination char[25];
  Language char[25];
  FatherName char [25];
  FDesignation char[25];
  MOthername char[25];
  MDesignation char[25];
  Phone_number number [10];
  DegreeClass char [30];
  Degree char[30];
  Area char[30];
  Address char[50];
  City char[30];
}
```

**Fig 4a: A student class**

- For example the student class from the campus management student database is vertically partitioned into V1, V2 & V3. The vertical partition V1 is partitioned vertically based on the personal details of the class student. Similarly partition V2 is partitioned based on the family details, and V3 is partitioned based on the
address. Partitioning agents are used to identify the related or dependent attribute and partition it appropriately in respective partitions. The following diagrammatic representation shows clearly how these attributes are partitioned:

4.2.2 Strategy and Algorithm Used

A unit of partitioning can be either method or an attribute. Attribute (method) as a unit of partitioning implies attributes (methods) are partitioned first and methods (attributes) are included in the partitions subsequently. They are method-based partitioning and attribute based partitioning. In the method based partitioning method, methods are partitioned first and the attributes are inserted afterwards. Whereas in attribute based partitioning the attributes are portioned first and the methods are inserted afterwards. In this research work, emphasis has been given only to attribute based partitioning as a beginning to vertical partitioning an Object Oriented Data set in an automated data mining system (i.e.) the related / closely dependent attribute are vertically partitioned to a class and methods are inserted afterwards.

The Figure 4b shows a partition set in which the unit is an attribute. The attributes of the class partitioned vertically based on the dependency factor. The dependency here refers how each attributes are related or closely related to one another. Each attributes are accessed in the class, to identify the attributes that are closely related or related to one another and their corresponding methods. For each attribute if there is a relation between the two then it is attached to the
existing class. New partitions are created if there are no related methods or attributes.

<table>
<thead>
<tr>
<th>Class V1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal Details</td>
</tr>
<tr>
<td>Registrernumber Char [10];</td>
</tr>
<tr>
<td>Initial Char [5];</td>
</tr>
<tr>
<td>FirstName char [20];</td>
</tr>
<tr>
<td>MiddleName char[20];</td>
</tr>
<tr>
<td>LastName char[20];</td>
</tr>
<tr>
<td>Dob char[20];</td>
</tr>
<tr>
<td>Gender char[5];</td>
</tr>
<tr>
<td>Country char [20];</td>
</tr>
<tr>
<td>Religion char[20];</td>
</tr>
<tr>
<td>Denomination char[25];</td>
</tr>
<tr>
<td>Language char[25];</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class V2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family Details</td>
</tr>
<tr>
<td>FatherName char [25];</td>
</tr>
<tr>
<td>FDesignation char[25];</td>
</tr>
<tr>
<td>Mothername char[25];</td>
</tr>
<tr>
<td>MDesignation char[25];</td>
</tr>
<tr>
<td>Phonenumber number [10];</td>
</tr>
<tr>
<td>DegreeClass char [30];</td>
</tr>
<tr>
<td>Degree char[30];</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class V3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Details</td>
</tr>
<tr>
<td>Area char[30];</td>
</tr>
<tr>
<td>Address char[50];</td>
</tr>
<tr>
<td>City char[30];</td>
</tr>
</tbody>
</table>

*Fig 4b: Possible Vertical Class partitions V1, V2 & V3*
4.2.3 Vertical Partitioning Algorithm

The following algorithm is a general algorithm for vertically partitioning an object-oriented database based on methods or attributes. First the user interface agent performs a detailed study about the object oriented database. The user interface agent analyze the number of attributes, number of classes, number of objects, and number of records of an object oriented data set keeps record of the database. And the analyzed information is communicated to the vertical
partitioning agent. Based on the analyzed information the vertical partitioning agent partitions the data set with the help of the partitioning processor. The vertical partitioning algorithm is given below.

**Vertical Partitioning Algorithm:**

```plaintext
Function VerticalPartitioning (Cv: set of classes to be vertically partitioned)
Ck is the set of attributes in a given class
Oi is the set of objects
returns Fv: set of vertical class partitions
begin
for each attribute Ck that is in Cv do
  for each attribute ci of ck that is accessed
    for each attribute cj of ck that is accessed by Qi do
      if there is a relation between ci and cj then
        link it into already existing class Fvi
      else
        create a class between ci and cj
        and partition the attribute into a new class Fvj
return Fv
end
```

**Figure 4d: Vertical Partitioning Algorithm**

The working of the vertical partitioning algorithms is as follows. Let Cv be the set of classes that is to be vertically partitioned. Ck is the set of attributes in the given class. In this research work the student class as shown in Figure 4a is the class to be partitioned. The first attribute from the class is read and based on
the user input the user interface agent see it beliefs which is a generic relational model specifically designed so that a belief set can be queried. The belief set has all the properties of the attributes. The two attributes are checked for the relationship between them. If there exists a relationship they are partitioned into a separate class as shown in figure 4b. Like that all the attributes from the given class is partitioned based separately based on the relationship till all the attributes are moved into one of the partitions.

4.3 Ranking Algorithm

The attributes in a partitioned data set are ranked in order of their relevance based on the query weight and scoring function to form a ranked list. The selection of attributes is carried out by ranking them according to the importance (weight). Ranking of attributes are essential in order to choose the correct attributes. Ranking agent helps in ranking the attributes. Ranking is performed by number of factors: Query weight and Scoring function. The highest ranked attributes are considered for clustering in this research work.

a) Query Weight:

The query set, Q is determined by the application programs that are run on the system.

Query set Q = \{q1, q2, q3, ..., qn\}

Attribute set A = \{a1, a2, a3, ..., an\}

\[ W_i = \sum r_a n_{i,a} \]
Query $Q_i$ occur $n_i$ a times in applications. "a," which is run $r_a$ times during the time period of interest. The greater the query weight, the higher the rank given to the attribute.

**b) Scoring Function:**

Based on the size and type of the attributes a Scoring function $F$ defined over a set of attributes $t$. $R$ assigns a ranking Score $F(R) [t]$ to each attribute $t$. Within each partition, the top $k$ attributes with the highest scores (or all if there are less than $k$ attributes in the cluster) are returned. The highest ranked scored attributes are considered for clustering.

The ranking agent will decide which ranking factor to choose either Query weight or scoring function. If the query has been already executed over the attributes it will choose the query weight. If no query has been executed over a particular attribute the scoring function will be chosen and the attributes are ranked. Because of the use of the ranking agent it will also allow the user to choose from any of the ranking factors mentioned above with little human intervention.

```
BEGIN
VARIABLES
  NUM% Number of predicting attributes in the Object Oriented data set
  TAR % Target Attribute from the given query
  RNK % List of predicting attributes sorted by decreasing relevance
  SEL % Selected attributes for clustering
INITIALIZATION
```
Figure 4e: Ranking algorithm with respect to an attribute in an object oriented data.

The ranking agent helps in ranking the attributes. And the highest ranked attributes are considered for clustering by the data mining agent. From the above algorithm the NUM is the number of predicting attributes in the object oriented data set. TAR is the target attribute from the given query. RNK is the ranked list generated by the ranking factors query weight and scoring function. The summation of the query weight and the scoring function gives the ranked list of attributes. For the highest ranked attributes run the clustering data mining algorithm based on the attributes. If the attribute is numeric the data mining agent will list and recommend the most suited algorithm with less human
intervention from any three algorithms used K- Means Algorithm or K- Medoids or PAM: Partitioning Around Mediods or CLARA: Clustering Large Applications. If the selected (SEL) attribute is a categorical attribute will lists K- Modes Categorical Clustering or Robust Clustering Algorithm (ROCK) and recommend the most suited algorithm with less human intervention. If the (SEL) selected attribute is an object oriented data then will list Cactis Algorithm or ORION Clustering Method for mining the data.

4.4 Clustering Algorithms

Cluster analysis is a technique used in data mining. Cluster analysis involves the process of grouping objects with similar characteristics, and each group is referred to as a cluster. Clustering is a useful technique for discovery of data distribution and patterns in the underlying data. The goal of clustering is to discover dense and sparse regions in a data set. An important class of problems in the areas of decision support and reporting are clustering. In this research work various clustering algorithms are identified based on the importance and performance of the algorithms over the different attributes.

A data set to be clustered contains a set of \( N \) Objects. An object \( o \) has \( m \) attributes, \( \{ o_1, ..., o_m \} \). Each attribute \( o_i \), \( i=1,...,m \), has a domain \( D_i \) of a data type, such as categorical or numerical. The research work uses various clustering algorithms.
Several clustering methods are identified for datasets with numeric, categorical, and object oriented attributes and considered in this research to implement automated data mining with software agents.

a) Algorithms for Numerical Attribute

- K- Means Algorithm
- K- Medoids or PAM: Partitioning Around Mediods
- CLARA: Clustering Large Applications

b) Algorithms for Categorical Attribute

- K- Modes Categorical Clustering
- Robust Clustering Algorithm (ROCK)

c) Algorithms for Object Oriented Data

- Cactis Algorithm
- ORION Clustering Method

4.4.1 Algorithms for Numerical Attribute:

k- Means Algorithm

k-means clustering is an algorithm to classify or to group your objects based on attributes/features into K number of group. K is a positive integer number. The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid. k-Means is a clustering dealing with numerical attribute values (NAs) primarily, although it can also be applied to categorical datasets with binary values, by viewing the binary values as numerical. The k-Means clustering algorithm for numerical datasets requires
the user to specify the number of clusters to be produced and the algorithm builds and refines the specified number of clusters [ARS, 98].

During the k-Means clustering algorithm for numerical datasets, the following is the steps performed:

i) Insert the first k objects into k new clusters.
ii) Calculate the initial k means for k clusters.
iii) For each object o
   a. Calculate the dissimilarity between o and the means of all clusters.
   b. Insert o into the cluster C whose mean is closest to o.
iv) Recalculate the cluster means so that the cluster dissimilarity between mean and objects is minimized.
v) Repeat 3 and 4 until no or few objects change clusters after a full cycle test of all the objects.

Each cluster has a mean associated with it. Means are used to choose the closest cluster to an object by computing the dissimilarity between the cluster's mean and the object. In the steps above, the object is then allocated to the closest cluster and the mean gets updated.

A dissimilarity metric is needed to choose the closest cluster to an object by computing the dissimilarity between the cluster's mean and the object. Assume that each object is described by m numerical attributes. Let \( X = \{x_1, x_2, \ldots, x_m\} \) be an object, where \( x_i \) is the value for the ith attribute, and \( Q = \{q_1, q_2, \ldots, q_m\} \) be the mean of a cluster. The dissimilarity between \( X \) and \( Q \) is defined as:
Dissimilarity \((X,Q) = \sum_{j=1}^{m} \frac{x_j - q_j}{m}\)

A mean \(Q\) for a cluster \(C\) with \(n_C\) objects is found by computing for each attribute position \(i\): \(\sum_{j=1}^{n_C} \frac{X_{i,j}}{n_C}\).

k-means is efficient with a computational complexity of \(O(tkN)\), where \(N\) is the number of objects, \(k\) is the number of clusters and \(t\) is the number of iterations.

The user needs to specify in advance for k-means the number of clusters \(k\). k-means is unable to handle noisy data and outliers and it is not suitable to discover clusters with non-convex shapes. Finally, the results depend on the order of the objects in the input dataset as different orderings will produce different results.
k - Medoids or PAM: Partitioning Around Mediods

The problem of outliers posed by k-means is dealt by k-Medoids. K-Medoids is similar to k-Means except that the mean of each cluster is the object that is nearest to the “center” of the cluster. [KR, 90]

The idea of partitioning with k-Medoids is to reduce the distance between all objects in a cluster and the most centrally located object in the cluster. The strategy is very similar to k-Means strategy. The steps are as follows:

i) First k of the N objects are inserted in k clusters arbitrarily.

ii) The medoid for each cluster is set equal to the object inserted in it.

iii) Each remaining object is inserted into the cluster whose medoid is most similar to it.
iv) Then each medoid \( I \) in a cluster may be swapped with one of the non-medoids \( h \), as long as the total swapping cost is negative.

The total cost for swapping medoid \( I \) with non-medoid \( h \) is determined by using the function: \( TC_{ih} = \sum_j C_{jih} \)

**CLARA: Clustering Large Applications**

K-medoids works effectively for small datasets but does not scale well to large datasets. CLARA (Clustering Large Applications) attempts to overcome this problem. It is an extension of k-medoids whose main focus is to scale well for large datasets. CLARA selects a sample of the entire dataset as a representative of the dataset. Medoids are then chosen from this sampling using a method similar to k-medoids. If the sampling has been done properly, the medoids chosen from the sample are usually similar to the ones that would have been chosen from the whole dataset. The effectiveness of CLARA depends on the size of the sample selected, since it searches for medoids among the selected sample.

**4.4.2 Algorithms for Categorical Attribute:**

**k-Modes Categorical Clustering**

The k-Modes categorical clustering algorithm requires the user to specify the number of clusters \( k \) as an input parameter. [ZH, 98], [ZH, 97] k-modes assigns a mode to each cluster as a summary of the cluster's most frequent
ats... bute values. The mode of cluster $c$ is a vector $\mu_c = \{\mu_{c1}, ..., \mu_{cm}\}$ where $\mu_{ci}$ is the most frequent value for the $ith$ attribute in $c$.

Given a dataset and a number of clusters $k$, k-Modes clusters the dataset as follows:

i) Select initial $k$ objects, insert each object in a new cluster and set each cluster's mode equal to its object's values.

ii) For each object $o$:
   a. Calculate the dissimilarity between $o$ and the modes of all clusters
   b. Insert $o$ into the cluster $c$ whose mode is the nearest to $o$.
   c. Update the mode of cluster $c$

iii) Retest the dissimilarity of objects against the current modes. If an object is nearer to the mode of another cluster rather than its own cluster, reallocate the object to that cluster and update the modes of both clusters.

iv) Repeat (3) until no or few objects change clusters after a full cycle test of all the objects.

A dissimilarity metric is used to choose the nearest cluster to an object, by computing the dissimilarity between the cluster's mode and the object. Let $o = \{o_1, ..., o_m\}$ be an object where $o_i, i=1..m$, is the $ith$ attribute's value. The dissimilarity between $o$ and $\mu_c$ is defined as:
\[ \text{dist}(o, \mu_c) = \sum_{i=1}^{m} \delta(o_i, \mu_{ci}) \]
where \( \delta(o_i, \mu_{ci}) = \begin{cases} 1, & \text{if } o_i \neq \mu_{ci} \\ 0, & \text{if } o_i = \mu_{ci} \end{cases} \)

Robust Clustering Algorithm for Categorical Attributes (ROCK)

ROCK is an adaptation of an agglomerative hierarchical clustering algorithm for categorical data. The user need not specify the number of clusters. ROCK assumes a similarity measure between attributes and defines a "link" between two attributes whose similarity exceeds a threshold \( \omega \) [SRK, 00]. Initially, each attribute is assigned to a separate cluster and then clusters are merged repeatedly according to the closeness between clusters. The closeness between clusters is defined as the sum of the number of "links" between all pairs of attributes, where the number of "links" represent the number of common neighbours between two clusters. The steps involved in clustering using ROCK are described below in figure 4g.

Data \( \xrightarrow{\text{Draw random sample}} \) Clustering with links \( \xrightarrow{\text{Label data in disk}} \)

Figure 4g: Overview of ROCK
**Figure 4h: Clustering Algorithm for ROCK**

ROCK accepts as input the set $S$ of $n$ sampled points to be clustered (that are drawn randomly from the original data set), and the number of desired clusters $K$. The procedure begins by computing the number of links between pairs of points in step 1. Initially, each point is a separate cluster. For each cluster $i$, build a local heap $q[i]$ and maintain the heap during the execution of the algorithm. $q[i]$ contains every cluster $j$ such that $\text{link}(i, j)$ is non-zero. The clusters $j$ in $q[i]$ are ordered in the decreasing order of the goodness measure with respect to $i$, $g[i,j]$. The goodness measure $g(C_i, C_j)$, that is,

$$\sum_{p_q \in C_i, p_r \in C_j} \text{link}(p_q, p_r).$$

Then, the goodness measure $g(C_i, C_j)$ for merging clusters $C_i, C_j$ as follows:
\[ g(C_i, C_j) = \frac{\text{link}[C_i, C_j]}{(n_i + n_j)^{1+2f(\theta)} - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}} \]

The pair of clusters for which the above goodness measure is maximum is the best pair of clusters to be merged. The pairs of clusters with a large number of cross links are, in general, good candidates for merging.

In addition to the local heaps \( q[i] \) for each cluster \( i \), the algorithm also maintains an additional global heap \( Q \) that contains all the clusters. Furthermore, the clusters in \( Q \) are ordered in the decreasing order of their best goodness measures. Thus, \( g(j, \max (q | j |)) \) is used to order the various clusters \( j \) in \( Q \), where \( \max (q | j |) \), the max element in \( q | j | \), the best cluster to merge with cluster \( j \). At each step, the max cluster \( j \) in \( Q \) and the max cluster in \( q[j] \) are the best pair of clusters to be merged.

The while-loop in step 5 iterates until only \( k \) clusters remain in the global heap \( Q \). In addition, it also stops clustering if the number of links between every pair of the remaining clusters becomes zero.

### 4.4.3 Algorithms for Object Oriented Data

**Cactis Algorithm**

Cactis is an object oriented multi user DBMS developed at the University of Colorado [HK, 89]. It is designed to support applications that require rich data modeling capabilities with object oriented features. The Cactis clustering algorithm is a static algorithm since it is periodically used to recluster the
database when the database is idle. This implies that the database is not clustered on the first run because no information about the database is available [CLM⁺, 93]. User’s hint is not required for this algorithm. This is an advantage since no arbitrary choice has to be made by the user. The time to compute total number of times each object is accessed and number of times each relationship is crossed needs more time.

**Algorithm**

```
Repeat
    Choose the most referenced object in the database that has not been assigned a block.
    Place this object in a new block.
Repeat
    Choose the relationship belonging to some object assigned to the block such that:
        1) the relationship is connected to an unassigned object outside the block and,
        2) the total usage count for the relationship is the highest.
    Assign the object attached to this relationship to the block.
Until the block is full.
Until all objects are assigned blocks
```

**Figure 4i: Cactis Algorithm**

In this research work with our student data let us say to cluster six objects into blocks of size 10. The object's relationship with each other is given by table 1. The size of each object, the number of times it has been accessed, lists of objects with which it is
related and the number of times each of these relationships has been accessed are shown below in the following table 1.

<table>
<thead>
<tr>
<th>Object name</th>
<th>Size</th>
<th>Number of times accessed</th>
<th>Relationships</th>
<th>Number of times crossed</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>4</td>
<td>75</td>
<td>O4</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>O5</td>
<td>70</td>
</tr>
<tr>
<td>O2</td>
<td>2</td>
<td>170</td>
<td>O3</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>O6</td>
<td>170</td>
</tr>
<tr>
<td>O3</td>
<td>6</td>
<td>40</td>
<td>O1</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>O2</td>
<td>60</td>
</tr>
<tr>
<td>O4</td>
<td>5</td>
<td>80</td>
<td>O4</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>O6</td>
<td>70</td>
</tr>
<tr>
<td>O5</td>
<td>3</td>
<td>50</td>
<td>O1</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>O5</td>
<td>100</td>
</tr>
</tbody>
</table>

**Figure 4j: Object characteristics for the clustering example of the student data set with Cactis**

Algorithm trace:

NEW BLOCK O5 selected
O5-O4 relationship selected, o4 selected, block full

NEW BLOCK O2 selected
O2-O5 relationship selected, O6 selected

NEW BLOCK O1 selected, all object clustered
ORION Clustering Method

ORION is a series of next generation database systems that have been prototyped at MCC (Microelectronics Computer Technology Corp). ORION is designed for Artificial Intelligence (AI), Computer-Aided Design and Manufacturing (CAD/CAM) and Office Information System (IOS) applications. ORION supports only a simple clustering scheme [KGB, 90].

ORION provides direct support for composite objects, i.e., objects with a hierarchy of exclusive component objects shown in figure 4g. The hierarchy of classes to which the objects belong is a composite object hierarchy is shown below for the student data set.

![Composite Object Hierarchy Diagram]

**Figure 4k: Example of composite object with the student dataset**

87
Procedure:

Step 1: Select the first object to cluster

Step 2: Get the whole composite hierarchy (if any) attached to this object

Step 3: Cluster all the objects belonging to this composite hierarchy into a new segment.

Step 4: Remove the objects belonging to this composite hierarchy from the set of objects to cluster

Step 5: Select the next object to cluster.

Step 6: Repeat from Step 2 until all the composite hierarchies are clustered.

Step 7: Cluster together by class the remaining objects into distinct segments.

Figure 41: ORION Algorithm

User assistance is required to determine which classes should share the segment. In ORION, segments have a fixed size. When a segment is full, a new page is allocated and linked to the segment. The advantage of this method is its simplicity that makes the method fast and easy to implement.
4.5 Data Base Used

Due to rapid advancement in the field of Information Technology, the amount of information stored in the educational databases is rapidly increasing. These huge databases contain a wealth of data and constitute a potential goldmine of valuable information. As, new courses and new colleges emerge in the environment the structure of the educational database changes. The clustering aspect of data mining offers comprehensive characteristics analysis of students, while the predicting function estimates the likelihood for a variety of their outcomes.

This research work uses real time 100k records from the Karunya University, Campus Management System database. It has 45 attributes of different data type. The data set uses numeric, categorical and date data type. The following are the list of attributes used in this research work. APMDOB, APM_PINCODE, APM_ADDRESS3, APM_ENTRANCE, APM_BLOOD, APM_PARENTTYPE, APM_INDIAN, APM_LATERAL, APM_DEL, APM_GENDER, APM_ADDRESS1, APM_COMMUNITY, APM_HOSTEL, APM_DOA, APM_DISTRICT, APM_TENTH, APM_CASTE, APM_MTONGUE, APM_BRANCH, APM_CUSER, APM_NAME, APM_DEGREE, APM_BATCH, APM_RELIGION, APM_CATEGORY, APM_SEMESTER, APM_DENOMINATION, APM_MNAME, APM_TWELVETH, APM_UGMARKS, APM_DIPLOMA, APM_FATHER, APM_LNAME, APM_ADDRESS4, APM_EMAIL, APM_TWELVETHMAJOR, APM_MDATE, APM_MUSER, APM_MOTHER, APM_PHONE1, APM_ADDRESS2, APM_FNAME, APM_CDATE.
The user interface agent analyzes the total number of attributes, number of classes, number of objects, and number of records from the given data set. In addition, it also identifies the relation between the objects, methods, classes, attributes etc.

4.6 Implementation

The system design is based on object-orientation and is implemented using JACK™ Intelligent Agents (JACK) is an Agent Oriented development environment built on top of and integrated with the Java programming language. It includes all components of the Java development environment as well as offering specific extensions to implement agent behaviour. JACK's relationship to Java is analogous to the relationship between the C++ and C languages. C was developed as a procedural language and subsequently C++ was developed to provide programmers with object-oriented extensions to the existing language. Similarly, JACK has been developed to provide agent-oriented extensions to the Java programming language. JACK source code is first compiled into regular Java code before being executed. The agents used in JACK are intelligent agents. They model reasoning behaviour according to the theoretical Belief Desire Intention (BDI) model of artificial intelligence.

Following the BDI model, JACK intelligent agents are autonomous software components that have explicit goals to achieve or events to handle (desires). To describe how they should achieve these desires, BDI agents are
programmed with a set of plans. Each plan describes how to achieve a goal under varying circumstances. Set to work, the agent pursues its given goals (desires), adopting the appropriate plans (intentions) according to its current set of data (beliefs) about the state of the world. This combination of desires and beliefs initiating context-sensitive intended behaviour which is part of the characteristics of a BDI agent.

A JACK agent is a software component that can exhibit reasoning behaviour under both pro-active (goal directed) and reactive (event driven) stimuli. Each agent has:

- a set of beliefs about the world (its data set),
- a set of events that it will respond to,
- a set of goals that it may desire to achieve (either at the request of an external agent, as a consequence of an event, or when one or more of its beliefs change), and
- a set of plans that describe how it can handle the goals or events that may arise.

When an agent is instantiated in a system, it will wait until it is given a goal to achieve or experiences an event that it must respond to. When such a goal or event arises, it determines what course of action it will take. If the agent already believes that the goal or event has been handled (as may happen when it is asked to do something that it believes has already been achieved), it does
nothing. Otherwise, it looks through its plans to find those that are relevant to the request and applicable to the situation. If it has any problems executing this plan, it looks for others that might apply and keeps cycling through its alternatives until it succeeds or all alternatives are exhausted. Agent communication is accomplished using KQML (Knowledge Query and Manipulation Language).

Consider an objective with the student database and see how the framework works with the case of object oriented data. Our objective is "Cluster the students based on the city". The object oriented data is analyzed by the User Interface Agent. The user interface agent follows the procedure as mentioned in the figure 5.1. It identifies the relationship between the other attributes. The objective specified city is related with pincode & district. The user interface agent after analyzing the data set, communicates the classified information to the partitioning agent.

The partitioning agent with the help of the partitioning algorithm, partitions the object oriented data. In this example 'pincode, district is partitioned separately. Once the data set is partitioned it calls the ranking agent to rank the attributes. The ranking procedure is given in fig 3.3. Based on the scoring function and query weight the attributes are ranked. In this case, pincode is a numeric attribute. Therefore, it takes the high scoring function. Data mining agent identifies and selects the appropriate algorithm. For this example it displays the following three numeric attribute algorithms. They are: k-Means,
k-Medoids and CLARA. With less user intervention the agent helps the user to select the appropriate mining algorithms. Once the user enters the number of clusters it follow the algorithm specified under 6.3.1 and the knowledge is mined for the given user hint or query.

4.7 Agent Communication

Agent Communication Languages (ACL) has been used in Multi Agent Systems for agents to exchange information and knowledge. The most widely known one is Knowledge Query and Manipulation Language (KQML) [FFM*, 94]. KQML (Knowledge and Query Manipulation Language) constitutes a good candidate to be used as a communication language between agents.

KQML is primarily concerned with pragmatics and, secondarily, with semantics. Pragmatics among computer processes includes knowing who to talk with and how to find them, as well as knowing how to initiate and maintain an exchange. KQML is a language and a set of protocols which support computer programs in identifying, connecting to, and exchanging information with other programs.