An Efficient Storage of Spatial Database using Nested Relational Database

Suchitra Reyya[1], M. Sundara Babu[2], Ratnam Dodda[3]
[1]*Assistant Professor, Lendi Institute of Engineering & Technology*
[2, 3]*Assistant Professor, P V P Siddhartha Institute of Technology*

Abstract

A spatial preference query ranks objects based on the qualities of features in their spatial neighbourhood. For example, using a real estate agency database of flats for lease, a customer may want to rank the flats with respect to the appropriateness of their location, defined after aggregating the qualities of other features (e.g., restaurants, cafes, hospital, market, etc.) within their spatial neighbourhood. Such a neighbourhood concept can be specified by the user via different functions. It can be an explicit circular region within a given distance from the flat. Another intuitive definition is to assign higher weights to the features based on their proximity to the flat. In this paper, we formally define spatial preference queries and propose appropriate indexing techniques and search algorithms for them based on IR-tree. Extensive evaluation of our methods on both real and synthetic data reveals that an optimized branch-and-bound solution is efficient and robust with respect to different parameters. There is also a hybrid index structure called Spatial-Tag R-tree (STR-tree), which is an extension of the R-tree for efficient spatial search. As the data representing in the R-Tree, IR-Tree and STR-Tree are in flat relation database (1NF), where this can occupy more amount of space to store. To give an efficient storage, we proposed nested relational database to reduce the space complexity and removal of redundancy in the flat relation database.

1. Introduction

The key to spatially enabling the enterprise lies with the Relational Database Management Systems (RDBMS). The RDBMS is the backbone of enterprise IT infrastructures. The RDBMS is used to store a wealth of mission critical information. However, while most of the organizations data tends to be stored centrally in the RDBMS, the spatial data still tends to be locked up in proprietary formats. In order for the enterprise to truly become spatially enabled a spatial database that adheres to open standards must be present. An open spatial database also enables all of the applications that connect to the database to take advantage of the spatial capabilities. However, traditionally non-spatial application such as a human resources application can also be easily modified to do things such as find the nearest co-worker to where an employee lives. While this is not one of the biggest areas that one would want to integrate spatial capabilities, it is a good example of what is possible once an enterprise's systems have been spatially enabled.

The Geographic databases (GDB) store real world entities, also called spatial features, located in a specific region. Spatial features (e.g. Belgium, Brazil) belong to a feature type (e.g. country), and have both non-spatial (e.g. name, population) and spatial attributes (geographic coordinates x,y). In GDB every different feature type is usually stored in a different relation, because most geographic databases follow the relational or object-relational approach. Figure 1 shows an example of geographic data representation using MAPs, where the spatial feature type's city, street, and near resources are different relations with spatial (shape) and non- spatial attributes.

The spatial attributes of geographic object types, represented by shape in Figure 1, have intrinsic spatial relationships (e.g. close, far, contains, intersects). Because of these relationships real world entities can affect the behaviour of other features in the neighbourhood. This makes spatial relationships be the main characteristic of geographic data to be considered for data mining and knowledge discovery. It is also the main characteristic which differs geographic/spatial data mining from non-spatial data mining. The rectangular shapes in the Figure 1, containing a relationship between city, street and near resources (eg., hotels, cinema theatres, hospitals, etc.,). The shapes containing geographic data will be stored in relational database, which shows in Table 1.

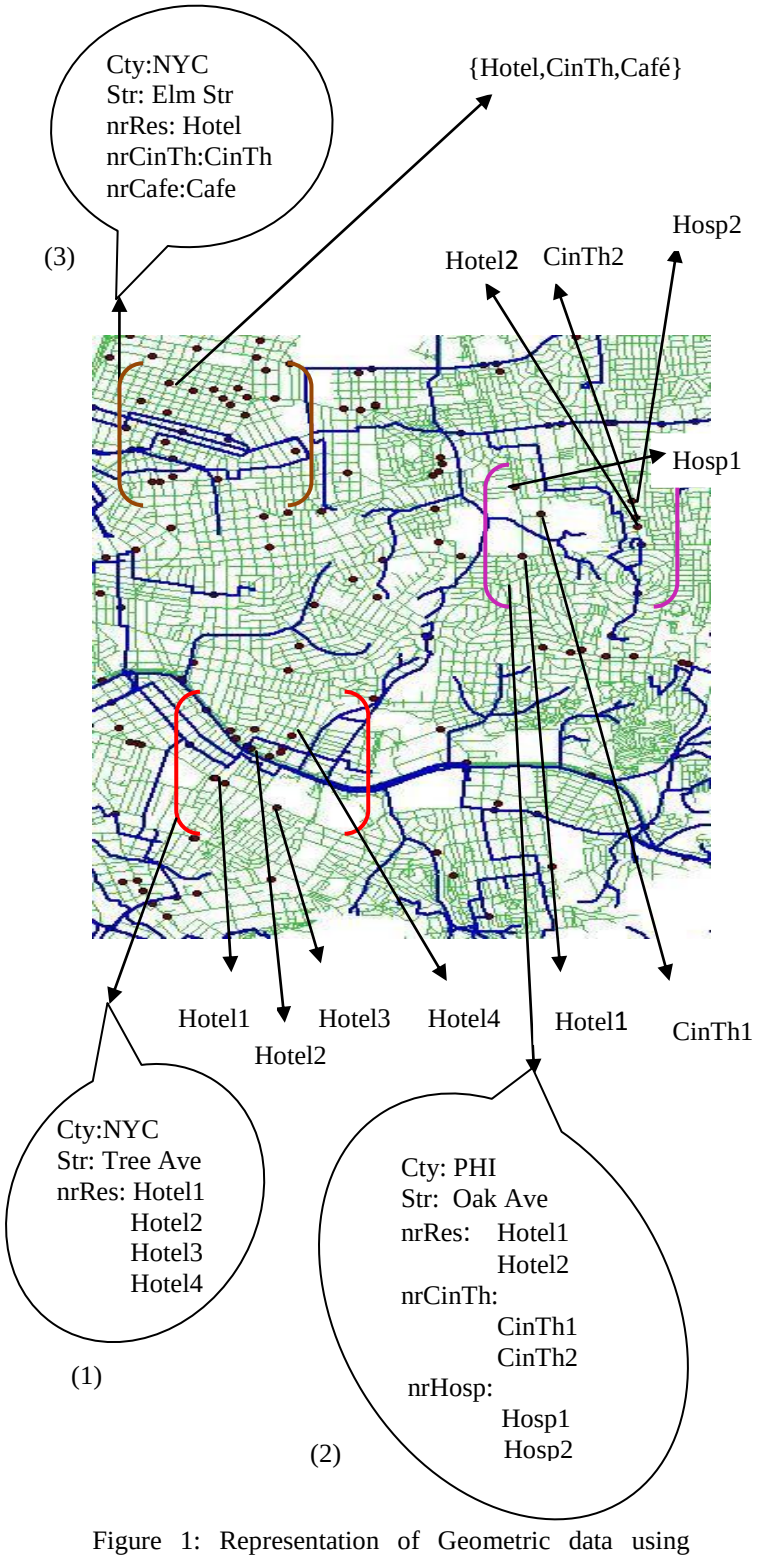

Figure 1: Representation of Geometric data using MAPs

The existing techniques are R-trees, IR-Tree, STR-Tree, which are the tree data structures used for spatial access methods, i.e., for indexing multi-dimensional information such as geographical coordinates, rectangles or polygons. These techniques generate databases which are in flat relation database (1NF) leads to an excess storage space on disk.

This paper introduces the flat relational database storage of R-Tree [1], IR-Tree [2] and STR-Tree [3] into nested relational database storage, which reduces the redundancy and space occupy by the disk.

The remaining paper is organized into sections, where section 2 discusses related work, section 2 introduces how to convert flat relation database into nested relation database and reducing storage space, section 4 explains methodology to reduce storage space and redundancy, section 5 deals with experimental analysis and section 6 concludes the paper.

2. Related Work:

Existing technique is the R-tree [1], which was proposed by Antonin Guttman in 1984 and has found significant use in both research and real-world applications. A common real-world usage for an R-tree might be to store spatial objects such as restaurant locations or the polygons that typical maps are made of: streets, buildings, outlines of lakes, coastlines, etc. and then find answers quickly to queries such as "Find all museums within 2 km of my current location", "retrieve all road segments within 2 km of my location" or "find the nearest gas station" [1]. The query of the R-Tree database storage is in such way that which gives the group of values for a single object. This is shown in the Table 1.

Table 1: Nearest group of values for Single Object (R-Tree)

CITY	STREET	Near Restaurant
NYC	Tree Ave	Hotel1
NYC	Tree Ave	Hotel2
NYC	Tree Ave	Hotel3
NYC	Tree Ave	Hotel4
NYC	Elm Str	Hotel1
NYC	Elm Str	Hotel2

R-tree extended with inverted files is known as IR-Tree. Here IR-tree's search for the nearest single object which is located in an area. The IR-Tree's leaf nodes contain entries of the form (p, p.λ, p.di), where p refers

to an object in dataset D, p.λ is the bounding rectangle of p, and p.di is the identifier of the description of p. Each leaf node also contains a pointer to an inverted file with the text descriptions of the objects stored in the node. Here Figure 2a contains 8 spatial objects p1, p2, . . . , p9, and Figure 2b shows the words appearing in the description of each object. Figure 3. illustrates the corresponding IR-tree, Recall that a joint top-k spatial keyword query Q consists of a set of sub queries qi. The arguments are a joint query Q, the root of an index root, and the number of results k for each sub query.

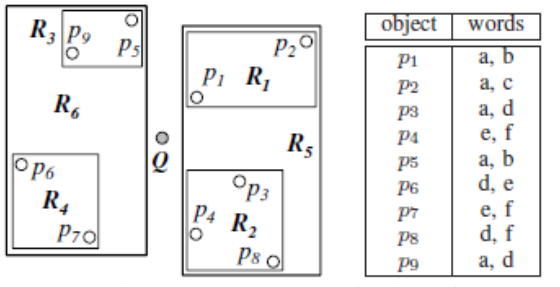

Figure 2a,b. Shows object locations and object descriptions

When processing a sub query qi € Q, which maintains a priority queue U on the nodes to be visited. The key of an element is the minimum distance mindist(qi.λ, e.λ) between the query qi and the element e. Later on utilizes the keyword information to prune the search space. Which only loads the posting lists of the words in qi. A non-leaf entry is pruned if it does not match all the keywords of qi. Then it returns k elements that have the smallest Euclidean distance to the query and contain the query keywords.

We want to find the top-1 object. Since entries R5 and R6 both contains a and b, both entries are inserted into the priority queue with their distances to q1. The next de-queued entry is R5, and the posting lists of words a and b in InvFile-R5 are loaded. Since only R1 contains a and b, R1 is inserted into the queue, while R2 is pruned. Now R6 and R1 are in the queue, and R6 is de-queued. After loading the posting lists of words a and b in InvFile-R6, R3 is inserted into the queue, while R4 is pruned. Now R1 and R3 are in the queue, and R1 is de-queued.

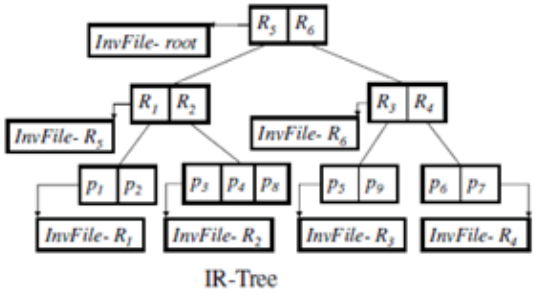

Figure 3. IR – Tree Structure

By using the minimum distance equation the IR-Tree [2] finds out the exact single object in an area basing on the human object location. Table 2 shows the nearest restaurants (single object) in an area with distance. By applying IR-Tree on Table 2, we can find the exact nearest distance (minimum distance) location which shown in Table 3.

Table 2: Nearest group of values for single object with distance

CITY	STREET	Near Restaurant	Distance (KM)
NYC	Tree Ave	Hotel1	5
NYC	Tree Ave	Hotel2	4
NYC	Tree Ave	Hotel3	6
NYC	Tree Ave	Hotel4	1
NYC	Elm Str	Hotel1	2
NYC	Elm Str	Hotel2	3

Table 3: Exact nearest value for a single object using minimum distance equation (IR-Tree)

CITY	STREET	Near Restaurant	Distance (KM)
NYC	Tree Ave	Hotel4	1
NYC	Elm Str	Hotel1	2

The above approaches aim to retrieve only single objects as query results. In contrast, to find group of objects such that the objects in a group collectively satisfy the needs of multiple users. This can achieve by using STR-Tree, which is a hybrid index structure called Spatial-Tag R-tree (STR-tree).

An extension of IR-Tree is a STR-Tree, where we can study how to find suitable spatial objects to meet multi users' needs in collaborative activity planning. We

formulate a new kind of spatial queries called Tag-based top-k Collaborative Spatial Query, which aims to retrieve top-k groups of objects for meeting users' needs. In essence, the spatial objects returned by a query should satisfy the following conditions: (1) they should be annotated with as many tags specified in the query as possible; (2) the objects in the result should be as close to one another as possible, such that the maximum diameter of the area covering the objects is minimized; (3) the maximum distance between the users' locations and the objects should be minimized.

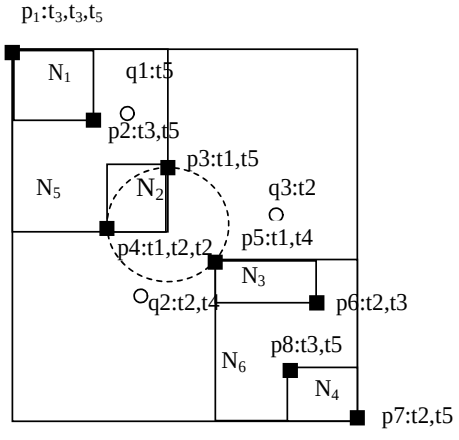

Figure 4. Explains the nearest multiple objects of multiple users to meet.

Figure 4 illustrates the query by an example. Points p1 ... p8 represent different spatial objects distributed in the region of Los Angeles, each object is annotated with a number of descriptive tags ti . Suppose three users, Bob, Tom and Mary, plan to find a place to meet. They collaboratively submit a query Q= {hRowenaAve : Moderat ePizzai, hMononSt : FreeParking, cinemai, hRussellAve : cinemai}, where Rowena Ave(q1) and Monon St(q2) and Russell Ave(q3) represent users' locations, and Moderate Pizza(t5), Free parking(t4), cinema(t2) are query tags that express their needs. As shown in Figure 2, the group of objects {p3 , p4 , p5 } appears to be the best choice, since it can (1) cover all the user's tags, (2) cover the smallest area, and (3) be near to the users' locations. In contrast, the objects {p2, p5, p6} are not suitable. Although they are optimal choices for some individual users (e.g. p2 for q1), not all the users' needs are well covered [3]. STR-Tree database is stored in flat relation, which shown in Table 4.

Table 4: Nearest values for multiple objects in an area (STR-Tree)

CITY	STREET	Near Objects
NYC	Tree Ave	Hotel
NYC	Tree Ave	Cinema Theatre
NYC	Tree Ave	Cafe
PHI	Oak Ave	Hotel1
PHI	Oak Ave	Hospital
PHI	Oak Ave	Free Parking
PHI	Oak Ave	Cafe
NYC	Elm Str	Hotel2
NYC	Elm Str	Shopping Mall

3. Reduction of Redundancy & Storage Space

This paper introduces multi-valued data in databases, which can give efficiency in data storage and elimination of redundancy.

Definition 1 (Multi-valued database): A multi-value attribute is the practice of maintaining more than a single value in a database column.

Our contribution is to convert first normal form (flat relational) database into nesting (nested relational) database using spatial database.

Definition 2 (First Normal Form): A relation is said to be in First Normal Form (1NF) if and only if each attribute of the relation is atomic. More simply, to be in 1NF, each column must contain only a single value and each row must contain the same number of columns.

Complex-valued data models can help us to overcome several limitations of relational data model in designing many practical data base applications. One of the complex-value data model is the nested relational model in which an attribute contains an atomic value or a nested relation.

Definition 3 (Nesting): Nesting is a fundamental operation for the nested relational data model, in which data tuples that have matching values on some fixed attribute set can be represented as a single nested tuple by collecting set of the different tuple values on the remaining attributes [4]. This concept can be explained by using Table 1 as input and result is shown in Table 5.

Table 5: Flat database conversion into Nested database

CITY	STREET	Near Restaurant
NYC	Tree Ave	{Hotel1, Hotel2,Hotel3,Hotel4}
NYC	Elm Str	{Hotel1,Hotel2}

The database in the Table 1 representing atomic information, but the data relation is in multi-valued and leads to redundancy and more storage space i.e., CITY: NYC and STREET: Tree Ave have representing more than one HOTEL as an unique hotel name. This gives the tuple wise data as a unique, and column wise data is in redundant. Our intention is to reduce the redundancy in column wise data. While reducing redundancy in column wise the database storage is also effects, shown in Table 5. Here Table 1 considering 6 tuples and Table 5 reduced those 6 tuples into 2 tuples by using flat relation to nested relation query.

4. Methodology:

This section includes a query to convert the given flat relational database to nested relational database to reduce redundancy and disk storage space.

Query. It converts Flat Relational database into Nested Relational database.

The concept of nesting is a useful notion in the study of non first normal form relations.

Let U be a set of attributes and R a relation over U. In this Relation an attribute will either be an ordinary attribute or a nested set of attributes. Let X, Y attributes of R and subsets of U.

Let $\Phi \neq X \subseteq U$ and Y=U-X

$$NESTx\ (U, R) = (Ux, Rx)$$

where Ux is a set of attributes, equal to (U-X).

Rx= {t| $\exists$ a tuple u ε π_y(R) such that t[Y] =u and t[X]={v[X]|v ε R and v[Y]=u}}.

Example

Consider the relation

(U={C, S, NO}, R) in Table 1.

$Nest_G$(U, R)=({C, S, NO' }, Rx)

where NO' is a multi-valued data of NR

Rx is a nested relation which is shown in Table 3

Suppose we have the relation schema Flat Relation with the attributes City(C), Street(S) and Near Objects (NO). The purpose of this database is to record the information of near objects for a particular city and street.

Table 6: Flat Relational database for near Objects

CITY(C)	STREET(S)	Near Objects(NO)
NYC	Tree Ave	Hotel1
NYC	Tree Ave	Hotel2
NYC	Tree Ave	Hotel3
NYC	Tree Ave	Hotel4
NYC	Elm Str	Hotel1
NYC	Elm Str	Hotel2
NYC	Tree Ave	Hotel
NYC	Tree Ave	Cinema Theatre
NYC	Tree Ave	Cafe
PHI	Oak Ave	Hotel1
PHI	Oak Ave	Hospital
PHI	Oak Ave	Free Parking
PHI	Oak Ave	Cafe
NYC	Elm Str	Hotel
NYC	Elm Str	Shopping Mall

The query cited below in the paper implements the one-nesting. It takes Table 6 as input and produces Table 7 as the output. We can clearly observe the elimination of redundancy in the attribute Near Objects (NO) after the Query result.

```
select distinct C,S,NO[(select NO from flattable
                        where C=f.C  and L=f.L
                        group by NO)] from flattable f
                        group by C,L;
```

Table 7: Nested Relational database for near Objects

CITY(C)	STREET(S)	Near Objects(NO)
NYC	Tree Ave	{Hotel, Hotel1, Hotel2, Hotel3, Hotel4, Cinema Theatre, Cafe}
NYC	Elm Str	{Hotel, Hotel1, Shopping Mall}
PHI	Oak Ave	{Hotel1, Hospital, Free Parking, Cafe}

5. Experimental Analysis:

In this section, we present our findings about the performance of our schemes for reducing redundancies over generalized database.

Setup: For the experiments, we used postgres SQL on Windows XP with 1.86 GHz Power PC dual CPU with 3GB of RAM.

Data: Our experiments used Adult Database (UCI) with a few synthetic attributes addition.

There are two alternative evaluation strategies for the comparison of flat and nested relations. 1. Graph representation for a time variation. 2. Graph representation for a size variation.

Graph representation for a Time variation: We studied the impact of reduced redundancy in the nested relation by varying time. In this we considered number of tuples whose value varied from 100 to 700 tuples in 100 increments can be observed in Table 8 and Figure 5.

Graph representation for a Size variation: In this experiment we investigated how the redundancy is reduced by considering the space occupied by the flat and nested relations on the disk by varying number of tuples can be seen in Table 9 and Figure 6.

Table 8:. Time for Query Execution

No.of Tuples	Flat Relation Time(msec)	Nested Relation Time(msec)
100	0.605	0.157
200	1.249	0.19
300	1.659	0.225
400	2.218	0.27
500	3.061	0.33
600	3.242	0.357
700	4.3	0.37

Table 9: Size of the relation

No.of Tuples	Flat Relation Size(KB)	Nested Relation Size(KB)
100	16	8
200	24	9
300	32	9.5
400	40	10
500	48	16
600	56	20
700	64	15

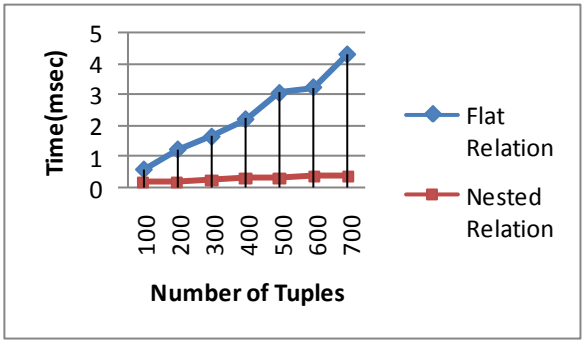

Figure 5. Graph representing Query Execution time

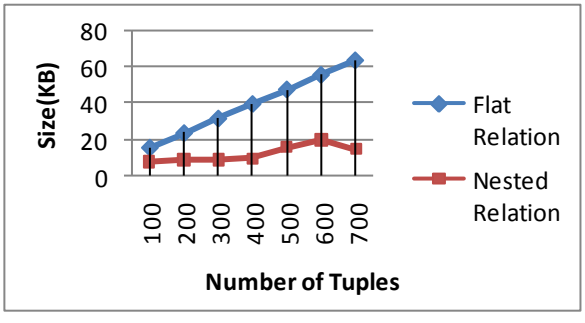

Figure 6. Graph representing Size of the relations

6. Conclusions:

We presented that, the spatial databases in nested relational database proved to be good in eliminating redundancy in generalized databases. We considered a single attribute for nesting and showed the experiments resulting in space and time saving of nested relation over a flat relation. There is naturally much more to be done. First, to remove redundancy on fly databases (incremental databases), second, applying nesting on different unstructured and semi structured data and third, to extend the nesting of multiple attributes and to verify with different attribute combinations.

7. References:

[1] Antonin Guttman, "R-trees: a dynamic index structure for spatial searching" *Proceeding SIGMOD '84*, Proceedings of the 1984 ACM SIGMOD international conference on Management of data pp 47 - 57

[2] D. Hari Krishna, Ch. Sowjanya, P. Radhakrishna, "Quality Preferences by using H.2.4.k Spatial Databases", IJCST Vol. 3, Issue 1, Spl. 5, Jan. - MarCh 2012

[3]Jinzeng Zhang, Xiaofeng Meng, Xuan Zhou, Dongqi Liu, "Co-Spatial Searcher: Efficient Tag-based Collaborative Spatial Search on Geo-Social Network", *Database Systems for Advanced Applications*, Lecture Notes in Computer Science Volume 7238, 2012, pp 560-575

[4] Korth, H., Roth, M.: "Query languages for Nested Relational Databases", Nested Relations and Complex Objects in Databases, Springer, (1991).
[5] Edward P. F. Chan, Rupert Zhu, "QL/G - A Query Language for Geometric Data Bases*", Proceedings of the 1st International Conf. on GIS in Urban Regional and Environment Planning*, pp 271–286, Samos,
[6] Hartmann, S., Link, S: "Characterising nested database dependencies by fragments of propositional logic*", Journal of Science direct*, Annals of Pure and Applied Logic 152 (2008) pp 84–106.
[7] G. Jaeschke and H.J. Schek, "Remarks on the algebra on nonfirst normal form relations", *Proceedings of the First ACM SIGACT-SIGMOD Symposium on the Principles of Database Systems*, pp 124-138, 1982.

Authors Biography

Suchitra Reyya received her B.Tech degree from Gayatri vidhya Parishad, JNTUH, Andhra Pradesh, India, in 2008 and the M.Tech degree from Pydah College of Engineering, JNTUK, Andhrapradesh, India, 2011. She was an Assistant Professor, in the Department of Computer science and Engineering, VITAM Engineering College. She was an Assistant Professor in the Department of Computer Science & Engineering, Pydah College of Engineering & Technology. Currently working as an Assistant Professor in the Department of Computer Science & Engineering, Lendi Institute of Engineering & Technology. She has 3 years of experience in teaching. She published one International Journal, two International Conferences. She is heading special interest research groups in Web technology, Oops through Java, Data mining.

M.SundaraBabu completed his B.Tech., M.Tech. He was worked as an Assistant Professor in Computer Science and Engineering, Pydah College of Engineering & Technology. Currently working as an Assistant Professor in Information Technology in PVP Siddhartha Institute of Technology. He had 3 years of experience in teaching. He published one International Journal. His area of interest is in Data Mining and Computer Networks.

D.Ratnam completed B.Tech.,M.Tech and currently working as a Assistant Professor in Information Technology in PVP Siddhartha Institute of Technology He had 3 1/2 years teaching experience. His area of interest is in Data ware housing and Data Mining and Image processing.