An Automated Forecasting Tool (AFT) achieved by clustering Entity Relationship Model

Preeti Mulay
Bharati Vidyapeeth University
Pune

Dr. Parag Kulkarni
Capsilon
Pune

Summary
We have described an algorithm (AFT) for decomposing ERDs into various modules / clusters, forming various tables equivalent to number of clusters generated and forecasting based on available raw data and generated clusters. Unlike earlier efforts, our algorithm clusters not only the entities but also the relationships involved. While designing this tool we have thought of its usability to all team members, application domain and extensibility to today's distributed systems as well. As of now this algorithm is fully automatic, works only on data available in the form of entities, relationships, tuples, fields, feature vector etc. It identifies suitable entity and relationship clusters without any further human (subjective) intervention.

The next phase of this algorithm (AFT) will involve human intervention also. One of the corporate companies have given one important feedback. They have one separate department in their organization made up of very knowledgeable and experienced team members only. Their aim is to learn customer thoroughly and think from customer's business point of view. This will give a value added services to customer. To achieve the same we may add, compare / contrast "expert's opinion" also.

We have discussed how our algorithm AFT, fulfills a comprehensive set of criteria for a good decomposition of ERDs. Our algorithm produces a more cohesive set of clusters while keeping inter-cluster coupling small. Our solution also offers a higher degree of modularity than that offered by other algorithms' solutions. While our algorithm produces very good solutions, it cannot guarantee their global optimality.
Our forecasting module of this proposed algorithm AFT will definitely prove to be most useful and suitable for all corporate teams, thereby saving their precious time which can be utilized on some other important chores.

Keywords
Databases; Information systems; Analysis and designs; Planning; Decision analysis, forecasting, clustering, entities, ERD, ERM

Abstract

Entity–relationship (ER) modeling is basically a technique of creating model before the detail database is formed. One the ERM or Entity Relationship Diagram (ERD) is complete and correct then it is feasible to convert into database design directly. The database design will also be complete and correct, as it is based on verified ERM and ERD. To increase the usability and understandability of ERM, ERD and large project database, I am planning to vote for classification of one of these or all these. There are two choices available: one is clustering based on this ERM or cluster database directly. I am planning to provide the algorithm based on clustering based on ERM / ERD, apply clustered details to database records / tuples and then forecast based on the available knowledge.

However, the complexities inherent in large ER diagrams have restricted the effectiveness of their use in practice. It is often difficult for end-users, or even for well-trained database engineers and designers, to fully understand and properly manage large ER diagrams. Hence, to improve their understandability and manageability, large ER diagrams need to be decomposed into smaller modules by clustering closely related entities and relationships.

Previous researchers have proposed many manual and semi-automatic approaches for such clustering. We are proposing an automated algorithm, not only for clustering large database or related ERD also extended to providing forecasting based on the same.

Our automated algorithm facilitates the re-clustering of ER diagrams as they undergo many changes during their design, development, and maintenance phases, as and when required.

The validation methodology used in this study considers a set of both objective and subjective criteria for comparison.

Introduction

Every organization or business firm need to store data in the most preferred form, and that is database. Pre-requisite required for preparing detail database is formation of entity relationship model and entity relationship diagram. ERM and ERD is a detail graphical representation of data requirement in an organization. This pictorial representation simulates details useful for database designers, engineers and complete team members. Processing the data available in the form of ERM or ERD or database will generate required knowledge.

Manuscript received December 5, 2008
Manuscript revised December 20, 2008

ERDs enhance understanding of the system and improve communication among database engineers, designers, and end-users.

Consider the following business scenario documented during systems analysis for the development of an information system for a retailer of, for example "book":

"A customer issues a enquiry to the vendor (retailer) to buy a book or few books for any library. The enquiry consists of an order item and book details. When the book is delivered, the customer is given a receipt for the book purchased including copies, title, author, edition, publisher. Because the customer has a line of credit, the actual payment is deferred until later. In the next billing cycle, the vendor issues an invoice that includes an invoice item related to the enquiry item on the purchase order. Upon receiving the invoice, the customer makes sales payment against the receipt of the book / books and the vendor receives vendor payment for the invoice."

This scenario results in the identification of the entities and relationships presented in a clustered
ERD presented in Fig. 1a. Entities are represented by rectangles, relationships by diamond-shaped boxes, and connecting lines show which entities participate in which relationship [1]. For example, the fact that a customer (Entity A) buys a book (Entity D) is represented by the ''buy'' relationship (Relationship 1).

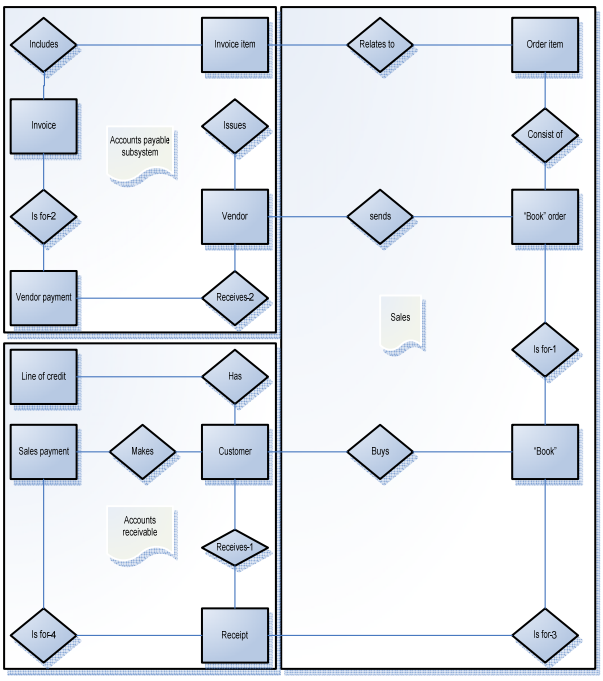

Figure 1 showing ERD for an information system of a retailer, for example "book":

In this example, weak or subtype entities are "order item" and "invoice item". Strong or super type entities are "book /s order" and "invoice".

Based on problem stated by customer, finalize ERM and ERD. One this is done database implementation becomes easy. There are various ER diagram generation tools available. This includes Visio by Microsoft, ER/1, Easy ER, etc. In addition to this there are various CASE tools available to achieve the same and much more. For ex. Turbo Analyst is the one which implements ERD and Data Flow Diagrams also. As mentioned here, there are various commercially / professionally available tools to help team members at various levels of Software Development Life Cycle (SDLC). But no single tool is available which can give complete simulation about predication based on all phases of SDLC. In this phase of our research we are not including human or expert's judgment at all. But we may add this feature later on.

Most ER tools support systems analysis and database design, implementation, and maintenance. Yet, today ER tools fall short of their true potential. This is because ER diagrams are rarely as small as the one presented in Fig. 1a. A typical application data model consists of 95 entities and an average enterprise model consists of 536 entities [2].

To improve the understandability and user friendliness, it is essential not only to classify / decompose / cluster ERM, but also retain knowledge in the form of tables for future use. This research paper focuses exactly on this, so as to generate most effective, efficient and useful tool for all levels of team members.

The ERD in Fig. 1a is small enough to comprehend without any decomposition. However, the three clusters (retailer's ''sales,'' ''accounts receivable,'' and ''accounts payable'' subsystems) identified by our algorithm illustrate some of the advantages of decomposing ERDs. The literature on clustering [3–6] has identified several advantages of ERD decomposition. The advantages of clustering ER diagram include:

- More readability and clarity
- Ensure completeness and correctness of the system
- More better organized
- Modulation of software tasks becomes easier
- Can develop easy and simple test cases based on the same
- Clusters can be stored separately in the form of tables for future reuse

- Forecasting, one of the major part of our research, will be achieved better with the use of these formed clusters.

Today no ER diagram gives complete clustering assistance even if many clustering advantages are known. This can be because of required intuitive and subjective judgment required from experts at various level during clustering algorithm's implementation.

ER tools support data model construction, communication, and validation by storing all the entities, relationships, relevant assumptions, and constraints in a repository. Using the repository, multiple conceptual and physical level ERD ''views'' can be produced for specific purposes such as end-user communication, database design, and development by presenting only relevant portions of the larger design to specific audiences. Users often prefer such views and better to provide such views to users from security point of view. In large organizations we need to consider the fact of overlap clusters. As many associated table will form some relationships based on requirements, it is essential to consider overlapping of clusters originally and at the time of forecasting also. ''Views'' may not offer some of the advantages of a clustered ERD.

We are planning to cluster ERD first, then the actual data and then form, store and reuse these clusters whenever the new project arrives. Next important issue is: not always the new projects will reuse already formed clusters; the feature set of new project may require generating new cluster. We have taken care of this requirement / possibility also. Generating these new clusters will help new projects, as the clusters set will be updated with new entries. These newly entered clusters also can be made available for forthcoming projects.

Manual algorithms which are available, developed by other researchers may work fine for small ERDs. However, for typical real-life (and large) ERDs, they are not likely to produce a good solution that seeks several clustering objectives simultaneously. As some of the researchers who has suggested similar methods for example, Moody and Flitman point out, ''Because of the enormous number of decompositions that are possible in even small data models, it is clearly beyond human cognitive capabilities to find an optimal solution. To handle this, our future work includes addition of expert's judgment along with provided forecast by our proposed algorithm.

Other algorithms which are available are time consuming. In today's demanding e-world we need to be the first and innovative in market and can't afford to spend time on stage like clustering. This will not help the ever changing / dynamic requirements from customers.

The ability to rapidly change databases and their underlying data models to support the needs of changing business models is high on the research agenda of information systems researchers [8].

In spite of their limitations, manual methods might have produced satisfactory decompositions of traditional transactional processing systems involve static and localized databases. However, new applications (from E-commerce to web-based decision support systems and from multimedia to geographic information systems) demand integrated databases with more entities, relationships, attributes, data elements, etc. The added semantics and complexity require that database engineers and designers revisit their conceptual data models periodically in an attempt to expand or modify them.

Furthermore, in software engineering, emphasis has shifted from a rigid system design, where the software is available at the end of the process, to an incremental and modular design based on iterative clustering refinements [10]. Clearly, an automated clustering algorithm can make iterative refinements considerably simpler and faster. According to Francalanci and Pernici
[5], in legacy database re-engineering, automated clustering is particularly useful due to the size of the existing physical schema.

Hence, it is no surprise that recent work in ER clustering has aimed at automated algorithms. Ideally, given the information obtained by database engineers and designers during the construction of an ERD, an automated algorithm should identify suitable entity and relationship clusters, without further human intervention. Unfortunately, none of the above referenced works meets that idea.

Clustering problems arise in numerous domains ranging from traffic analysis, weather forecasting, in hospitality industry, cellular manufacturing (CM), linguistics, data mining, and economics. In different contexts, clustering takes different names such as typology, numerical taxonomy, decomposition techniques and partitioning. Clustering methodologies originate in equally diverse disciplines including neural networks, genetics, fuzzy sets, matrix manipulation, mathematical programming, and multivariate analysis [15].

ERD clustering also require focus on various levels and hierarchy of entities along with various types involved. The number of relationships every entity participates in is better criteria for ERD clustering, as suggested by Flitman and Moody. Focus on validation, integration and reuse of schema is also important.

Related work

To achieve automated procedure, identifies suitable entity and relationship clusters without any further human intervention and subjective judgment we need to look at related work performed by other researchers.

The first attempt of automate clustering was by Zoeller and Huffman[13]. It is based on set of rules, find major entities and with the help of expert system.

One semi-automatic method was suggested by Pernici and Francalanci [5], based on one entity forms one cluster. After which most closed clusters will be combined together. Myself along with my guide we have formulated three proposals for "closeness factor" [30] among clusters and it work wonders.

To organize or decompose ERD, Martin suggested "hierarchical leveling". Clusters were formed with root entities as centers and their descendents as other elements in the cluster. Overlapping hierarchy chains were resolved by judgment from experts.

Teorey [6] suggested identifying functional area first before clustering entity. He achieved this by handling various entity types (and grouping them) including strong, weak, subtype, super type, binary and tertiary entities. All of these methods depended heavily on human judgment to resolve boundaries, to define strength of association, and/or to identify suitable subject areas.

Another way of ER clustering as suggested by Moody[20] is again, based on "entity and participation in various relationships". This connectivity between entities can give better idea about hierarchical chains overlap, and boundaries. Moody and Flitman [14] presented both a manual method and an automatic genetic algorithm method for ER clustering based on connectivity.

Criteria for proposed clustering solutions

Many researchers including those listed in reference at the end, have discussed about various methods of clustering ERD, clustering of Databases and forecasting separately. Hence we are proposing, and implementing, a combined algorithm, to achieve all the above and much more. Additional features as mentioned earlier include verification based on two different types of databases having relationships implemented successfully.

Before we implemented this algorithm, we did survey of many software development organization to get the feel about usability of our proposed tool. And the feedback was enormous. The best part of our algorithm will be: any organization (if wish to) can commercially implement this algorithm, by concentrating on their own and respective customer's quality issues.

One of our important observations include: practicing database engineers and designers want to see non-redundant clusters. At the same time users who must deal with multiple functional areas of a system often prefer to see overlapping database views. If some entities and relationships are relevant to the sales function and the accounting function, users prefer those entities to be shown in both, the sales view as well as the accounting view.

Let me throw light on this by giving you one more working example. We are working on one more interesting type of network called "Car Area Network (CAN)". This is nothing but handling all necessary functionalities within the boundaries of automobile car. This may include showing details regarding air pressure in all tires, gas availability in tank, source and destination details, GPRS system details, malfunctioning of any of the electrical / electronic and mechanical parts available in that new / updated / innovative / technology savvy automobile. To implement all requirements of CAN, the developer and the whole team must have knowledge about all clusters, overlapping views etc.

We agree that a user understandability may be served better by overlapping views, and clustered ERDs cannot be substitutes for functional views. On the other hand, non-overlapping clusters better serve a database designer's understanding and work allocation.

We are however at this phase are not considering the size of clusters formed. Right now the focus is developing and reusing clusters.

A typical cluster would have to involve 3 or 4 entities with their associated relationships. As described by Francalanci and Pernici [5] for purposes of reuse and schema validation, clusters should fit their task domains regardless of their size. Whenever we will focus on cluster size we may prefer "reasonable" size. We may also focus not only reusing clusters but also combining clusters based on requirements.

Another criterion proposed by Moody and Flitman [7], suggests that all the entities in a cluster should relate to a single central concept; and two entities, each involved in a large number of relationships, should be in different clusters.

Thus, the first part of our proposed algorithm is designed to meet the following criteria:

- Focus on completeness: in software development projects in particular, completeness is one of the mandatory functionality. Every requirement of customer need to be taken care in project. Hence the clustering applied on customer's data also need to be achieved completely. Top down and bottom up approaches if applied on database based cluster set should give the same results. No data should remain untouched.
- Focus on correctness: while performing clustering based on ERM / ERD first and then on actual data to generate knowledge, consider all fields and records correctly.
- Redundancy: every tuple and its relationship need to a part of only one cluster.
- User friendliness: formed clustered ERD, data and generated knowledge should be useful and must be represented in friendly way.
- Cohesion and coupling: all rules related to these two important functionalities need to be satisfied. Cohesion says: all entities within a cluster should be closely related to each other. And coupling needs all entities in different clusters should not be closely related to each other. Otherwise we will spend time resolving all dependencies.
- Formation of clusters: totally depends upon requirement.

As mentioned earlier we may opt to reuse already formed clusters and keep updating cluster set if required.
However, simultaneous attainment of maximal cohesion and minimal coupling is impossible. Since every entity is directly or indirectly connected to one or more entities, invariably, when decomposition increases within cluster cohesion, it also increases inter-cluster coupling. This is because, when each cluster contains a large number of entities and relationships, many entities are only indirectly connected to one another. Consequently, each cluster is not very cohesive.

At the same time, since the total number of entities and relationships outside a cluster is relatively small, very few entities are connected directly to entities outside their clusters. Thus, inter-cluster coupling is also small.

In previous research papers we have found the details given on this cohesion and coupling issue: In a properly clustered ERD, as the average cluster size decreases, since there are not as many indirect connections between entity pairs within each cluster, within-cluster cohesion increases. At the same time, inter-cluster coupling increases since now there are many more entities outside each cluster that may be directly related to the entities in that cluster.

In addition, our algorithm requires a table, in which lists all the weak and subtype entities in the ERD along with their corresponding strong or super-type entities and respective relationships.

The clustering algorithm

The complete algorithm is given below along with the required GUI's in following sections. The detail steps of this algorithm are explained in annexure.

1) Input Mo and To associated with ERM / ERD to be classified.
2) Remove weak and subtype entities and their connecting relationship from Mo to M1.
3) Remove singular entities and their connecting relationship from M1 creating M2.
4) Compute the distance between each pair of entities in M2.
5) Rearrange the rows in M2 to overlap pairs of entities with smallest distance creating M3.
6) Identify N, the largest number of potential clusters to consider.
7) For K = 2 to N
    a. Identify all S admissible sets of K groups of entities in M3
    b. For f = 1 to S
        i. Reinsert all data from M3 to create M4
        ii. Create M5, K-cluster solution from M4
    c. Next f
    d. Calculate G and identify best K-cluster solution
8) Next K
9) Identify optimal solution
10) Get the database entries in the form of tuples / records once the clustering is achieved
11) Store these clusters in the form of various tables, name them. Inform user about number of clusters generated and table details.
12) Compare feature vector (vector having complete feature set of new project) details with these clusters.
13) If match is found, then reuse these clusters, learn from these clusters and forecast.
14) If mismatch found, then store those mismatched features in the form of new cluster for future reuse.

Figure 2 shows complete clustering algorithm.

Forecasting or estimation tool framework

Once the clustering details are known then we need to move ahead with forecasting details. The following steps are a part of GUI generated as an estimation result of our proposed algorithm.

We have not fully implemented this result window but working on the same or better views. The result window will look like this including various valuable thoughts and knowledge.

Following Figure shows Framework for result window.

This framework gives details including the tile of "NEW" project which arrived at an organization (lets take an example of software development project to understand this framework). The next important estimated input based on available information about that project is how many clusters can be reused and how many "NEW" clusters will be required to form. One this estimation is ready then user will get to know links where this reusable and suitable information is available (including reusable diagrams, documents, classes / objects etc). Based on historical data available and clustered, algorithm can now estimate how much time this "NEW" project will take to complete. Next important suggestions will be about required team size, skills of team members, technology platform required, resources, cost, risk etc. All this estimation details prior to starting of "NEW" project will give better, clear picture in front of complete team which will ease development efforts drastically. Giving such forecasting was feasible only because of availability of historical software project data, and of course clustering techniques mentioned and used.

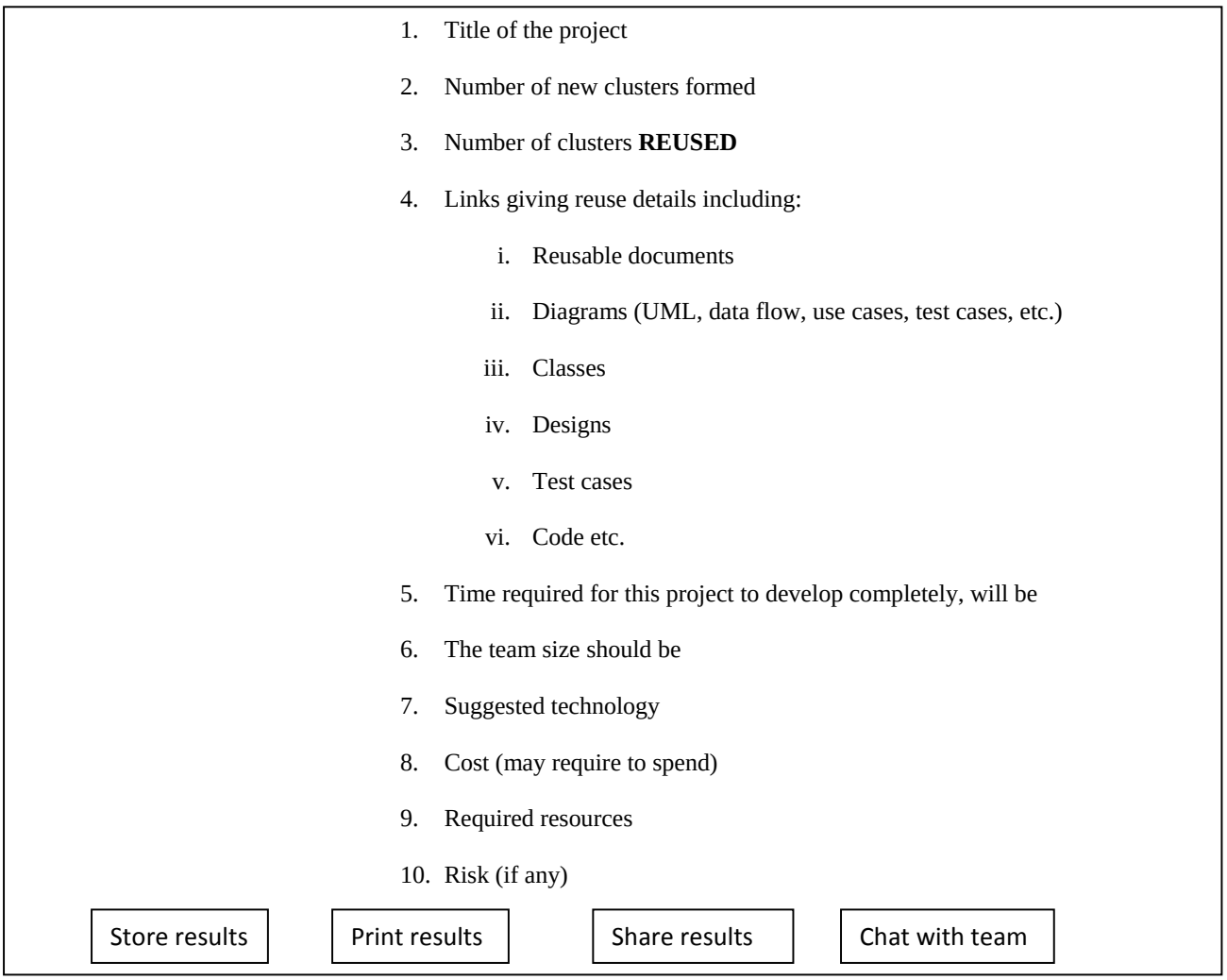

Figure 3 shows framework for estimation tool.

References

[1] P.P. Chen, The entity–relationship model—towards aunified view of data, ACM Trans. Database Syst.

[2] R. Maier, Benefits and quality of data modeling—results of an empirical analysis, in: B. Thalhiem

[3] P. Feldman, D. Miller, Entity model clustering: structuring a data model by abstraction, Comput. J.

[4] J. Akoka, I. Comyn-Wattiau, Entity–relationship and object-oriented model automatic clustering, Data Knowl. Eng. (1996)

[5] C. Francalanci, B. Pernici, Abstraction levels for entity–relationship schemas, in: P. Loucopoulus

[6] T.J. Teorey, G. Wei, D.L. Bolton, J. Koenig, ER model clustering as an aid for user communication and Documentation in database design, Commun. ACM 32 (8) (1989)

[7] D. Moody, A. Flitman, A methodology for clustering entity relationship models—a human information processing approach, in: J. Akoka, M. Bouzeghoub, I. Comyn-Wattiau, E. Metais

[8] M. Genero, G. Poels, M. Piattini, Defining and validating metrics for assessing the maintainability of ERD.

[9] A. Badia, Entity–relationship modeling revisited, ACM SIGMOD Manage. Data Notes 33 (1) (2004)

[10] C. Francalanci, A. Fuggetta, Integrating information requirements along processes: a survey and research directions, ACM SIGSOFT Software Eng. Notes 22 (1) (1997)

[11] S. Castano, V. De Antonellis, M.G. Fugini, B. Pernici, Conceptual schema analysis: techniques and applications, ACM Trans. Database Syst. 23 (3) (1998)

[12] P. Jaeschke, A. Oberweis, W. Stucky, Extending ER model clustering by relationship clustering, in: R. Elmasri, V. Kouramajian, B. Thalheim

[13] S.B. Huffman, R.V. Zoeller, A rule-based system tool for automated ER model clustering, in: F.H. Lochovsky

[14] D. Moody, A. Flitman, A methodology for decomposing entity relationship models, human information processing approach.

[15] M. Halkidi, M. Vazirgiannis, Y. Batistakis, Clustering algorithms and validity measures, Proceedings of 2001

[16] D.L. Moody, A.R. Flitman, A decomposition method for entity relationship models: a systems theoretic approach.

[17] J. Akoka, I. Comyn-Wattiau, Framework for automatic clustering of semantic models, in: R. Elmasri, V. Kouramajian, B. Thalheim

[18] J. Martin, Strategies for Data-Planning Methodologies, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[19] L.J. Campbell, T.A. Halpin, H.A. Proper, Conceptual schemas with abstractions—making lat schemas more comprehensible, Data Knowl. Eng. 20 (1) (1996)

[20] D.L. Moody, Entity connectivity vs. hierarchical leveling as a basis for data model clustering: an experimental analysis, Database Expert Syst. Appl. 2736 (2003).

[21] G. Biswas, J. Weinberg, C. Li, ITERATE: A Conceptual clustering method for knowledge discovery in databases, *In Innovative Applications of Artificial Intelligence in the Oil and Gas Industry*, B. Braunschweig, R. Day (Ed.), 1995..

[22] M.P. Chandrasekharan, R. Rajagopalan, An ideal seed nonhierarchical clustering algorithm for cellular manufacturing.

[23] A.J. Vakharia, U. Wemmerlov, A comparative investigation of hierarchical clustering techniques and dissimilarity measures applied to cell formation problem, J. Oper. Manage. 13 (2) (1995) .

[24] J. Miltenburg, W. Zhang, A comparative evaluation of nine well-known algorithms for solving the cell formation problem in group technology, J. Oper. Manage. 10 (1) (1991)

[25] P. Joglekar, M. Tavana, S. Banerjee, A clustering algorithm for identifying information subsystems, J. Int. Inform.Manage.

[26] C. Mosier, J. Yelle, G. Walker, Survey of similarity coefficient based methods as applied to the group technology configuration problem, Omega 25 (1) (1997).

[27] H. Seifoddini, A note on the similarity coefficient method and the problem of improper machine assignment in group technology applications, Int. J. Prod. Res. 27 (2) (1989).

[28] P. Joglekar, Q. Chung, M. Tavana, Note on a comparative evaluation of nine well-known algorithms for solving the cell formulation problem in group technology, J. Appl. Math. Decis. Sci. 5 (4) (2001).

[29] S.M. Ng, On the characterization and measure of machine cells in group technology, Oper. Res. 44 (5) (1996).

[30] Generalized Forecasting technique based on Pattern Matching (GFPM), Dr.Parag Kulkarni and Preeti Mulay, submitted to International Journal for Internet and Enterprise Management, special issue on "Advancing Software and Knowledge Engineering in a Connected World",2008

[31] An automated entity–relationship clustering algorithm for conceptual database design, by Madjid Tavanaa,, Prafulla Joglekara, Michael A. Redmondb, Management Department, La Salle University, Philadelphia, PA 19141, USA, Department of Mathematics and Computer Science, La Salle University, Philadelphia, PA 19141, USA

Dr. Parag Kulkarni is Ph.D. form IIT, Kharagpur (www.iitkgp.ernet.in). He is working in IT industry for more than 13 years. He is on research panel and Ph.D. guide for University of Pune, BITS and Symbiosis deemed University. He has conducted 5 tutorials at various international conferences and was a keynote speaker for three international conferences. He has also worked as a referee for International Journal for Parallel and Distributed Computing, IASTED conferences. He is also member of IASTED technical committee of Parallel and Distributed Computing. Presently he is General Manager-Technical at Capsilon India, Pune. He is also Honorary Professor at AISSM Engineering College, Pune and on board of studies for a couple of Institutes. He has worked as Senior Manager (R&D) at Scientific Applications Center, Siemens information systems Ltd., Pune. His areas of interest include image processing, security systems, decision systems, expert systems, classification techniques, load balancing and distributed computing.

Prof. Preeti Mulay is working on her PhD in the areas of "software engineering". She completed her MS (Software Engineering) from Wayne State University, MI, USA 2002 and M.Tech (Software Engineering) from JNTU, India, 2000. She is working in the education field since 1995,on various positions.

Other relevant reference articles are:

[Este96] M. Ester, H-P. Kriegel, J. Sander, X. Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, *In proceedings of the Second Knowledge Discovery and Data Mining conference*, Portland, Oregon, 1996.

[Gib00] D. Gibson, J. Kleinberg, P. Raghavan, Clustering Categorical Data: An Approach Based on Dynamical Systems, VLDB Journal 8 (3-4) pp. 222-236, 2000.

[Gant99] V. Ganti, J. Gehrke, R. Ramakrishnan, CACTUS - Clustering Categorical Data Using Summaries, *In Proc. Of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, San Diego, California, 1999.

[Gibs98] D. Gibson, J. Kleinberg, P. Raghavan, Clustering Categorical Data: An Approach Based on Dynamical Systems, *In Proc. of the 24th International Conference on Very Large Databases*, New York, 1998.

[Haim97] I.J. Haimowitz, O. Gur-Ali, H. Schwarz, Integrating and Mining Distributed Customer Databases, *In Proc. of the 3rd Int'l Conf. on Knowledge Discovery and Data Mining*, Newport Beach, California, 1997.

[Han96] J. Han, Y. Fu, W. Wang, J. Chiang, W. Gong, K. Koperski, D. Li, Y. Lu, A. Rajan, N. Stefanovic, B. Xia, O.R. Zaiane, DBMiner: A system for Mining Knowledge in Large Relational Databases, *In Proc. of the 2nd Int'l Conf. on Knowledge Discovery and Data Mining*, Portalnd, Oregon, 1996.

[Han01] J. Han, M. Kamber, Data Mining Concepts and Techniques, Morgan Kaufmann Publishers, 2001.

[Jain88] A.K. Jain, R.C. Dubes, *Algorithms for clustering data*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[Kett95] A. Ketterlin, P. Gancarski, J.J. Korczak, Conceptual Clustering in Structured Databases: a Practical Approach, *In Proc. of the 1st Int'l Conf. On Knowledge Discovery and Data Mining*, Quebec, Montreal, 1995.

[Ng94] R.T. Ng, J. Han, Efficient and Effective Clustering Methods for Spatial Data Mining, Proc. 20th Int. Conf. on Very Large Data Bases, Santiago, Chile, pp. 144-155, 1994.

[Nish93] S. Nishio, H. Kawano, J. Han, Knowledge Discovery in Object-Oriented Databases: The First Step, *In Proc. of the AAAI-93 Workshop on Knowledge Discovery in Databases (KDD-93)*, Washington, 1993.

[Quin93] J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

[Ribe95] J.S. Ribeiro, K. Kaufmann, L. Kerschberg, Knowledge Discovery from Multiple Databases, *In Proc. of the 1st Int'l Conf. On Knowledge Discovery and Data Mining*, Quebec, Montreal, Canada, 1995.

[Ryu98b] T.W. Ryu, Discovery of Characteristic Knowledge in Databases using Cluster Analysis and Genetic Programming, *Ph.D. Dissertation*, Department of Computer Science, University of Houston, Houston, 1998.

[Ryu98c] T.W. Ryu, C.F. Eick, Similarity Measures for Multi-valued Attributes for Database Clustering, *In the Proc. of the Conference on SMART ENGINEERING SYSTEM DESIGN Neural Networks, Fuzzy Logic, Evolutionary Programming, Data Mining and Rough Sets (ANNIE'98)*, St. Louis, Missouri, 1998.

[Ryu02] T.W. Ryu, W-Y. Chang, Customer Analysis Using Decision Tree and Association Rule Mining, *In the Proc. of the International Conference on SMART ENGINEERING SYSTEM DESIGN: Neural Networks, Fuzzy Logic, Evolutionary Programming, Artificial Life, and Data Mining (ANNIE'02), ASME press*, St. Louis, Missouri, 2002.

[Sala00] H. Salameh, Nearest-neighbor clustering algorithm for relational databases, *Master of Science Thesis*, Department of Computer Science, California State University, Fullerton, 2000.

[Shek96] E.C. Shek, R.R. Muntz, E. Mesrobian, K. Ng, Scalable Exploratory Data Mining of Distributed Geoscientific Data, *In Proc. of the 2nd Int'l Conf. On Knowledge Discovery and Data Mining*, Portland, Oregon, 1996.

[Wils97] D.R. Wilson, T.R. Martinez, Improved Heterogeneous Distance Functions, *Journal of Artificial Intelligence Research* 6,1997.

[Zhan96] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient database clustering method for very large databases, *In Proc. of ACM-SIGMOD Int. Conf. On Management of Data*, Montreal, Canada, pp. 103-114, 1996.

[Zehu98] W. Zehua, Design and Implementation Tool to Extract Structural Information from Relational Databases, *Master of Science Thesis*, Department of Computer Science, University of Houston, Houston, 1998.

Annexure: the detail clustering algorithm explained, *as suggested by Tavana, Joglekar etc*.

We have referred to clustering algorithm by Tavana, Joglekar [31]. Because of their great work, and similarity in our research build our ideas of on top of it. Based on their clustering details, we will detail the required forecasting in coming sections.

Starting with the M0 and the T0 associated with an ERD, our algorithm consists of three major

Parts presented in Fig. 1 shown above and described in detail below.

These important vital parts include: clustering ERM / ERD, then storing / forming tables based first part, the third and last most important is "forecasting".

We have started implementing this algorithm using java specially .net technologies. In our implementation, we identify each cell by its row and column names (tuple and field) and cells containing '1's are shaded while cells containing '0's are not.

A ''problem,'' an ER matrix to be clustered, can be created by choosing the number of rows and columns in the matrix, then clicking on cells which should contain '1's. Converting the ERD into its corresponding M0 and producing the corresponding T0 are easy, mechanical, and auto able tasks, since they require no information that is not already contained in the ERD.

Alternatively, M0 and T0 could be constructed through appropriate interface with existing ERD tools. All the steps following this input of M0 and T0 are fully automated. That is, from this point on, no human intervention is needed before the computer produces the optimal solution.

Remove weak and subtype entities and their connecting relationships from M0 creating M1

Next step requires removing weak and sub type entities, so as to concentrate on strong entities first. This step is very essential and need concentration because the completeness and correctness criteria depend on this step. We need to form two separate groups, one for weak and subtype entities and other for strong entities. We will make sure to store them separately as they will not be ignored completely. Finally they are entities.

Using the information in T0, the computerized algorithm's first step is to remove from M0 all weak and subtype entities as well as the relationships connecting the weak to the strong and the subtype to the super type entities. In this step, the algorithm ensures that the respective strong/super type Entity is now connected to all of the other relationships that were originally connected to its weak / subtype entity. This procedure produces a new matrix, M1.

Remove singular entities and their connecting relationships from M1 creating M2

Now this step in algorithm is to identify and remove each singular entity along with its associated relationship with M1. Sometimes, when a singular entity is removed from M1, a previously nonsingular entity becomes a singular entity in the reduced matrix. This happens when a singular entity's connected entity is related to only one other entity. Hence, the process of identification and removal of singular entities is repeated until there are no singular entities remains.

At the same time concentrating on singular entities relationship is also important and in the order of their removal, a list of all the singular entities and their associated relationships is created, so that at the appropriate time, they can be reinserted in the matrix, in the reverse order of their removal. The resulting matrix is labeledM2. Clearly, if one or more weak, subtype, or singular entities are removed, M2's dimension, will be smaller than M0's dimension

Compute the distance between each pair of entities in M2

The next part of algorithm requires computing distance between each possible pair of rows (also called entities) in M2. The shorter the distance between a pair, the more appropriate it is to put the two entities in the same cluster, and the larger the distance, the more appropriate it is to put the two entities in different clusters.

Rearrange the rows in M2 to juxtapose pairs of entities with smallest distances creating M3

A matrix, M3, is constructed where we leave the relationship columns in the same order as in M2, but rearrange the rows so that pairs of entities with least distances are closest to each other. The algorithm compares all pair-wise row distances, and copies the pair of rows with the least distance next to each other in M3. Among the unused Rows, the algorithm finds the one with the smallest distance from one of the current edge rows. The identified row is added to M3 next to that edge row. This process is repeated until there are no unused Rows in M2. Thus, at this point, M3 represents the rows in M2, in an order such that the pairs of entity rows with the smallest distances are closest together.

Identify N, the largest number of potential clusters to consider

The next step of this algorithm is to find the range for K, the number of clusters to consider. As suggested earlier, every cluster must contain at least two entities. Hence, the

maximum number of clusters, N, is the integer value of the number of entities divided by two. We also assume that, when an ER matrix is to be clustered, a database manager is looking for at least two clusters. Therefore, this algorithm assumes that the desired number of clusters, K, is in [2, N].

Identify the ‘‘best’’ K-cluster solution for every possible value of K

This is one of the important parts of the algorithm with several sub-steps detailed below. First, for each value of K (2 to N), the algorithm identifies all admissible sets (say S) of K groups of entities. Next, for each one of the S sets for a given value of K, the algorithm carries out the sub-steps, creating S matrices that are labeled M4. Then, each one of the M4's is used to create a feasible and promising K-cluster solution. Once all S feasible and promising K-cluster solutions are created, calculate the goodness of fit, G, for each one of them. Among the S solutions, the solution with the largest value of G is identified as the best K-cluster solution.

Identify all S admissible sets of K groups of entities in M3

Since a matrix's own boundaries (i.e. the first and the last rows) also serve as the boundaries for the first and the last sub-matrices, respectively, to construct a partition resulting in K groups of entities (rows), we need K-1 other horizontal dividers between rows. It is desirable to draw the dividers between the K-1 pairs of adjacent rows that have the greatest distances. However, since a cluster must have at least two entities, divider locations immediately after the first row, or immediately before the last row, cannot be considered selectable. Initially, all other possible divider locations are considered as selectable. The pair of adjacent rows with the largest distance is found and the first divider is placed between those two rows.

Since an entity group must contain at least two entities, entity pairs involving the rows immediately adjacent (on either side) to a selected divider are no longer selectable. Among the remaining pairs of entities, the pair of adjacent rows with the largest distance is found and the next divider is placed between those two rows. This process is repeated K-2 times to find all but the last divider. Up until this point, ties in largest distances are broken arbitrarily by choosing the first of the tied divider locations.

For the (K-1)st divider, often, there are several selectable divider locations with the same and
largest distance between adjacent rows. Hence, the last divider is handled differently to ensure that arbitrary tie-breaking does not cause a better solution to be missed. Suppose that there are S
candidates for the (K-1)st divider. Together, the first K-2 dividers and one of the S candidates
produce an admissible set of K groups of entities in M3. Thus, there are S sets of K groups of entities to consider..

When, for a given value of K, it is impossible to select any admissible (K-1)st divider, that value of K is eliminated from further consideration.

Reinsert all singular, weak, and subtype entities and their connecting relationships in M3 to create M4

First, for each admissible set, f ε S, of K groups of entities, the algorithm constructs a new matrix M4 by reinserting, in the reverse order of their removal, each singular, weak, and subtype entity immediately above its respective connected, strong or super type entity. If necessary, a divider location is adjusted to ensure that the reinserted entity and its relevant entity are in the same group. All the relationships are reinserted as the last columns of the matrix and any altered relationship column '1's are adjusted back to their original entity rows. Thus, at the end of this procedure, for each one of the S sets, the corresponding M4 is a p X q matrix consisting of all the entities and relationships in M0 and for a given value of K, the total number of M4's is S.

Create M5, a feasible and promising K-cluster solution from a given M4

Now, the columns in a given M4 need to be rearranged so that each group of entities is clustered with suitable relationships. Our aim is to maximize the goodness of fit of the resulting K-cluster solution. To maximize G, one should cluster the largest possible number of relevant relationships with the smallest group of entities.

Hence, for each M4, this algorithm creates a blank matrix, M5, of the same dimensions (p X q) as M4. It copies into M5 all the entity (row) names in the same order as in M4. The chosen K-1 dividers are also copied in M5. Along with the matrix boundaries, the dividers help identify the K groups of entities in M5. Then, an iterative process of column rearrangement and cluster identification begins.

Each iteration involves first identifying the smallest group of entities (rows) that has not yet been put into a cluster. In case of a tie in the entity group size, the first group is selected. The algorithm considers each relationship that has not already been copied to M5. If a relationship has at least as many 1's in the rows of the selected group of entities, as it has in the rows of any of the other unclustered groups of entities, that relationship

column is copied to matrix M5 in the next available column spot. When consideration of all relationships is done, the copied relationships and the selected group of entities are declared as a cluster, and the algorithm moves on to the identification of the next cluster. When all K clusters are constructed,M5 represents the feasible and promising K-cluster solution resulting from a given M4.

As can be noted, this is a greedy procedure for maximizing the G-value of the solution. However, there is no guarantee that an exhaustive approach would not have found a clustering arrangement with a higher value of G for this set of K groups of entities. That is why we call this solution as a ‘‘feasible and promising’’ solution.

Calculate G and identify the best K-cluster solution

Once all S feasible and promising K-cluster solutions have been created, the automated algorithm calculates the goodness of fit, G, for each one of them. The solution with the largest value of G is identified as the best K-cluster solution.

Identify the optimal solution to a problem

Finally, all the best K-cluster solutions for the entire range of K’s (2 to N) are compared based on their G values, and the solution with the highest G is chosen as the ‘‘optimal’’ clustering solution. Although we use the word ‘‘optimal’’ to refer to the best of the best solutions for various values of K, it should be clear that we are not claiming that our solution is globally optimal. Our algorithm is a heuristic that employs a greedy approach. Thus, we use the word ‘‘optimal’’ simply to avoid the use of the awkward phrase, ‘‘best of the best K cluster solution.’’

Last and important step of clustering

While this algorithm is designed to consider all possible values of the number of clusters, K = 2 to N, we have also incorporated a special feature in it.

As N is a variable, we may assign any required number to this N and execute the said algorithm to achieve required results. The very important part of the same algorithm is to generate tables as equal numbers of generated clusters and then focus on forecasting.