Journal of Computer Science 1 (3): 376-380, 2005
ISSN 1549-3636

Automatic Discovery of Association Paths in Relational Databases Using Software Visualization

Haider Ali Ramadhan
Department of Computer Science, Sultan Qaboos University, P.O. Box 36, Muscat 123, Oman

Abstract: We introduce a visual framework for facilitating tasks associated with database maintenance and re-use. The prototype system embodying the framework is presented. The system utilizes various techniques and features of software visualization. The system supports visual displays of the database structure along with various implicit relationships found in it such as associations and path views. Information visualized is automatically extracted from the database schema. To assess the usefulness of the proposed framework in helping the programmers to quickly recognize path views among relations, an empirical evaluation was conducted. Results collected from the evaluation seem to support our hypothesis that the time required to manually recognize path views from the database schema is considerable and tends to increase as the depth between the relations increases. The evaluation also showed that by using our visual framework such time is negligible and tends to be static.

Keywords: Software Re-Engineering, Databases, Software Visualization, Visual Interfaces, Empirical Evaluation

BACKGROUND

Large relational databases are inherently complex. Their evolution over time may increase the difficulty of understanding them even more. Due to this evolution, their structures may degrade over time to an extent that the tasks involved in maintaining and re-engineering these databases could become very costly. Therefore, techniques that may aid the designers and engineers in the analysis of these information systems deserve special attention and research focus. The task of understanding the structure and design of a relational database system is a complex one. There are many dimensions of complexity, three important among which are:

* The overall structure of the database. This includes the set of relations, their attributes and the types of data from which the attributes may take values.
* Various links between relations. This includes the number of associations between relations and their implicit nature.
* Various direct and indirect paths among relations in the database.

To help the developers get clear understanding about the above three dimensions, we have developed a prototype system using software visualization framework [3, 6]. The system is able to visually display the structure, i.e. relations and attributes, of a relational database in a limited space. In addition, the system visually shows the implicit relationships among relations in terms of associations and path views. The visualization is organized in separate views, each dealing with one of the dimensions mentioned above. These views include the overall view, association view, detailed view and path view. All these views are automatically visualized from the specifications of the database and hence provide a general tool for visualizing any relational database. In summary, our aim is to use software visualization [2, 4, 8] features to facilitate re-engineering, maintenance and analysis of the structure and relationships found in existing relational databases.

Software Visualization [6, 8, 9, 10, 11] refers to the task of applying visual and graphical techniques to exhibit the static structure and the dynamic behavior of software systems. The main purpose of software visualization is to provide designers, engineers, programmers and users with visual aids to help them in understanding and analyzing the structure and behavior of a software. This task is achieved through abstracting low-level textual structures, i.e. code and data, into high-level visual representations, hence reducing the mapping and interpretation load.

Figure 1 shows the database composition of a small firm with 19 relations. The relations are drawn as colored dots, where the color gives the range of the attributes in a relation as per the criteria: relations with green color have attributes ranging from 1 to 5, those with blue color have attributes ranging from 6 to 10 and those with red color have attributes greater than 11. It is worth noting that these dots are randomly displayed in the view. To accommodate databases with hundreds of relations, the size of the dots is automatically reduced and the scroll bars are introduced. Generally speaking,

regardless of the size, any database can be visualized in this manner. When a relation is pointed by the mouse in the overall view, its description is displayed in the relational information view, shown in the lower left corner of Fig. 1.

This view provides the textual information of a relation as it is described in the database specification. By having the relations in one view and their description in another, we can squeeze hundreds of relations to be visualized in the overall view. This information changes as the mouse moves to another relation in the overall view.

The association view (shown in the upper left corner of Fig. 1) displays the associations which exist between relations. When a relation dot is clicked by the mouse in the overall view, the associations for that relation are displayed in this view.

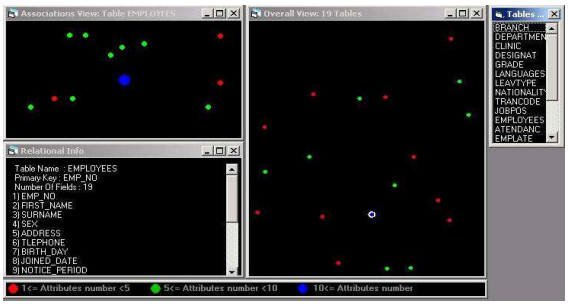

Fig. 1: Visual Representation of a Database

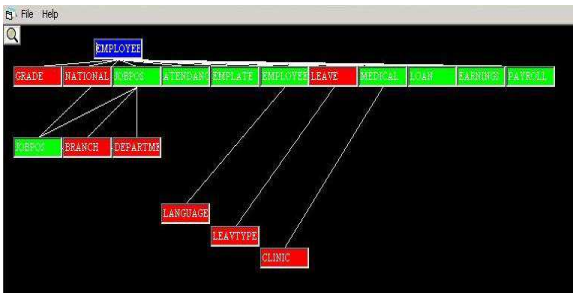

Fig. 2: The Path View

Figure 1 shows that the relation EMPLOYEES is associated to 11 other relations shown surrounding it. The relation selected is enlarged and placed in the center of the view. Relations associated to the selected relation can also be expanded to show their attributes and how these attributes are linked to attributes of the selected relation. Figure 2 shows the path view from the EMPLOYEE relation to all other relations in the database. A more improved approach would be to show paths between any two relations. However, to make the view clear only shortest paths in this case should be shown. It is hoped that the current implementation would still help in recognizing how relations are linked either directly or indirectly to form the association tree.

RELATED WORK

Traditionally, software visualization has been related to areas such as algorithm animation [7], program visualization [8] and computation visualization [5, 6]. In the area of program visualization, for example, visual techniques have been reported to reduce the complexity of low-level views of the source code and provide higher-level models of both static and dynamic behaviors as well as of the structural architecture of programs. By using different visual views such as call graphs, control flow graphs, data flow information, program slices and memory spaces, a software engineer can gain much clearer understanding of the program behavior and its functionality [7]. Some recent efforts have also managed to incorporate visualization features in the design and development of knowledge based systems [10].

However, when considering the importance of relational databases, work done in relation to database visualization does not seem to meet expectations. Here our task is to visually represent the complex structure of a relational database along with its behavior and various relationships, which are normally implicit and hence more difficult to manually detect. In other words, the challenge involved in visualizing the structure of a relational database is attributed to the fact that entities to be visualized are abstract, i.e. they have no physical form. Lack of enough visualization systems for relational databases may be attributed to this inherent difficulty. Recently, some work has been accomplished on visualizing the data stored in the database but not the structure [14]. In this type of work, graphical functions are provided to manipulate the data and accomplish what is normally done using SQL statements. Work on automated graphical presentation tool, APT, which provided static visual designs of relational information is regarded a seminal effort in this direction [12]. The focus of this system is on the visualization of formal characterization of semantic relational information. Like APT, our system works with minimal user input and supports a perspective mechanism for designing graphical representations.

Other systems were designed to visualize the logical model of the database through graphical presentation of the network view of the database schema [13]. Basically, here the focus is to visually represent E-R diagrams. Two main shortcomings of this approach can be outlined. First, E-R diagrams represent the logical model of the database using node and link graphs. Though being a novel technique for making explicit the logical design which is only implicit in the specifications of the database, the mapping between the E-R diagram and the relations in the database is not a trivial task even with visual E-R diagrams. In addition, visual E-R diagrams would not provide the software engineers with enough understanding of the overall

structure of the database nor with detail information at the attributes and associations level. Second, large relational databases with many relations would result in visual graphs which are bushy and cluttered, hence making the task of clearly understanding the structure and relationships not so easy. Therefore, visual E-R diagrams would be useful for small databases only.
Visualization of relational information in commercial database packages may be considered an improvement. However, issues related to scale and information coordination are two apparent pitfalls. For example, Oracle through graphical schema builder does support visual representation of associations among relations. However, visual displays of the associations and path views among relations tend to become bushy and difficult to understand for even a medium size database. Besides, finding out all associations to a single relation is not supported. We strongly believe that more efficient visual models are needed to provide better visualization of relations than the one provided by Oracle. We also believe that visual metaphors supported by our design provide more coherent representations and cater for simple views of larger databases. It would be interesting to formally support our claims through future empirical evaluations. Of course it would be worth noting that through the use of CASE based tools, most of the problems facing the designers of the legacy database systems could be overcome. Unfortunately, not all the institutions in this part of the world utilize such tools.

EVALUATION

We have planned several experimental evaluations to get some insights into the usefulness of our visual framework. These experiments involved 50 programmers and analysts who were asked to discover the implicit relationships in a medium-size database using both manual and automatic approaches. The manual approach involved recognizing the relationships using the database schema, while the automatic approach involved using our visual system. Both performances were critically compared to figure out the usefulness of our system. The first experiment focused on direct associations among relations. Results compiled from the evaluation strongly suggested the superiority of using our visual framework in quickly recognizing the associations [1]. The second experiment, reported in this study, focused on the discovery of indirect associations, namely the path views, among relations of the database. The purpose of this experiment was to find out how much manual effort in terms of time is put by a group of relatively experienced programmers to find various path views found in the structure of a relational database.
A total of 50 users took part in this investigation. The users were junior programmers drawn from IT departments of private and public institutions. Programmers selected had a job experience of 12 to 18 months and all were involved in tasks related to analysis, development and maintenance of applications programs which also included database applications. The users were randomly divided into two groups, each consisting of 25 users. These groups were named Visual and Manual. The Visual group used our prototype visual system while the Manual group used the textual print out of the database schema. We used the database of a small firm shown in Fig. 1. Both groups were asked to answer a total of eight questions dealing with path views of different depths.
Our hypothesis was that the time needed to identify path views among two relations would considerably increase as depth increases. In fact, results analyzed below strongly supported our hypothesis. Only one key performance measure was considered for evaluating and analyzing the results, namely the total time spent on ansewring these questions. The measure (speed of solution) aimed at finding out the time taken to determine these path views.
To avoid any subjective interpretation of the results, it was decided to assign a single point to every correct answer to a question. Responses which were not correct, including those which were close to the correct answer, were assigned zero. We admit that this approach is by no means represents the best criteria to measure the understanding of the users in regard to the questions asked. It is possible that users who failed to identify correct solutions for some of the questions did have a reasonably clear conceptual understanding of the database and its relations. We recognize this pitfall but still feel that this point-based approach in evaluating repsonses given by the users does give us some objective insights into the usefuleness of the visualization framework tested.
Figure 3 and Table 1 summarize the overall performance of both groups in terms of the time spent on answering these eight questions. The Visual group spent a total of 71 minutes on these questions while the Manual spent a total of 692 minutes. The mean for the Visual group is 2.84 minutes and for the Manual group is 27.68 minutes. This implies that the users in the Visual group spent an average of 21 seconds on each question, while the users in the Manual group spent an average of 3.46 minutes (207.6 seconds) on each question.
The difference in means suggests better performance of the Visual group. Overall, the Manual group using the textual version of the schema took 621 minutes (10.35 hrs.) longer to answer all nine questions than the Visual group. In terms of averages, this implies that the users in the Manual group spent on average 88 seconds more on each question than the users in the Visual group. Table 1 also shows the relationship between each question and the depth. For example, Q1 deals with finding path view of depth 2 while Q6 deals with finding a path view of depth 4.

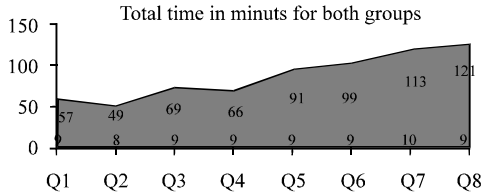

Fig. 3: Total Time Spent by Both Groups

Table 1: Total Time in Minutes for Both Groups

Questions	Man. Group	Vis. Group	Depth
Q1	57	9	2
Q2	49	8	2
Q3	69	9	3
Q4	66	9	3
Q5	91	9	4
Q6	99	8	4
Q7	113	10	5
Q8	121	9	5
Time	692	71	
Mean	27.68	2.84	

Table 2: Performance in Relation to Path Depths

Depth	2	3	4	5
Man	106/4.24	135/5.40	190/7.60	234/9.36
Vis	16/0.64	18/0.72	17/0.68	19/0.76

Table 2 shows the performance of both groups on questions grouped in relation to the depth of the path views. For example, the Manual group spent a total of 106 minutes on questions dealing with path views of depth 2 (mean = 4.24), while the Visual group spent a total of 16 minutes (mean = 0.64). This difference implies that the Manual group took 90 minutes longer to answer these two questions. For questions dealing with path views of depth 4, the Manual group spent a total of 190 minutes on them (mean = 7.60), while the Visual group spent a total of 17 minutes (mean = 0.68). This difference implies that the Manual group took 173 minutes longer to answer these two questions. These results seem to support the hypothesis we stated above which implies that the time required to recognize path views tends to increase as the depth between the relations increases. As shown by the table, the time to recognize path views jumped from 106 minutes for depth 2 to 135 minutes for depth 3, to 190 minutes for depth 4 and finally to 234 minutes when depth was set to 5. However, for the Visual group the time does not seem to increase much with the depth.

To see what is involved in finding path views from the schema, let us consider a path view of depth 2. Here the user needs to locate the first relation and then follow the *references* (associations) found in this relation one by one until the target relation is found. For example, if the first relation contains three references, then in the worst case the programmer would end up looking at all three relations referenced in the first relation. Using the system, the user (1) clicks on the first relation in the table list box so that the system automatically highlights the relevant dot in the overall view, (2) double clicks on the highlighted dot in the overall view so that the system shows the associations for that relation in the association view and finally (3) right clicks in the association view to display the viual representation of the path view. These few clicks is all what it takes to find a complete path view. Our results show that the users in the Visual group took on average 21 seconds to perform this task. No major increase in time as depth increases for the Visual group may be attributed to the efficient visualization supported by the system. The users seemed to have aquianted themseleves with this simple 3-step based clicking process regardless of the depth.

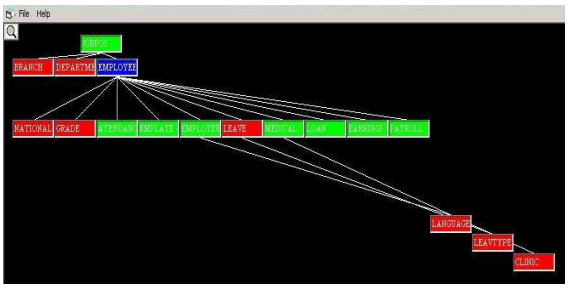

Fig. 4: A Path View of Depth 3

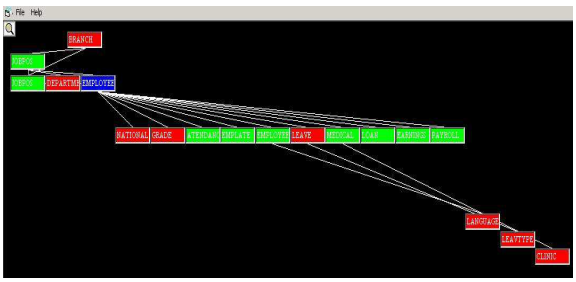

Fig. 5: A Path Veiw of Depth 4.

These results came as no surprise to us. The task of manually trying to find various associations among relations from the database schema is no trivial operation. For example, to find all associations for relation x, the user needs to do the following:

* locate the relation x in the schema
* identify the direct associations tagged under the references command in relation x and
* scan the entire schema to find relations that reference relation x, i.e. indirect associations for x.

Obviously, the time needed to find assocations following the above steps is co-related with the size of the database structure and hence the length of its

schema. With visual display, all the associations are displayed in colored circles on the screen and the programmer needs only to count these circles. When finding path views, the manual process would become even more cumbersome and errorprone.

CONCLUSION

We have introduced a visual framework to facilitate tasks associated with database maintenance and re-use. The prototype system embodying the framework provides graphical views of the database structure along with various implicit relationships found in it such as associations and path views. Information visualized is automatically extracted from the database schema. To assess the usefulness of the proposed framework we have conducted a series of experiments. The evaluation reported in the study focused on the discovery of indirect associations, namely path views, among relations of the database. The goal of this experiment was to find out how much manual effort in terms of time is put by a group of relatively experienced programmers to manually find various path views exist in the structure of a relational database at different depths. The time was compared to the one needed by the programmers to find out the same path views using our visual system. Results collected from the evaluation seem to support our hypothesis that the time required to manually recognize path views from the database schema is considerable and tends to increase as the depth between the relations increases. The evaluation also showed that by using our system such time is negligible and tends to be static.

REFERENCES

1. Ramadhan, H. and H. Al-Lawati, 2003. Design and Evaluation of a Visual Framework for Facilitating Re-engineering and Re-use of Relational Databases. Intl. Conf. Automation and Information, Spain, pp: 342-349.
2. Shneiderman, B., 1996. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations, Proceedings of the IEEE Conference on Visual Languages, pp: 336-343.
3. Card, S.K., P. Pirolli and J.D. Mackinlay. The Cost-of-Knowledge Characteristic Function: Display Evaluation for Direct-Walk Dynamic Information Visualizations, in Readings.
4. Stuart Card, Jock Mackinlay and Ben Shneiderman, 1999. Information Visualization: Using Vision to Think, Morgan Kaufmann.
5. Eick, S.G., 1998. Maintenance of Large Systems, Chapter 21 of Software Visualization: Programming as a Multimedia Experience by John asko, Blaine A. Price, Marc H. Brown (Edr), MIT Press.
6. Baker, M. and S. Eick, 1995. Space Filling Software Visualization. J. Visual Languages and Computing, 6: 119-133.
7. Ball, T. and S. Eick, 1996. Software Visualization in Large, IEEE Computer, pp: 33-43.
8. Jerding, D. and J. Stasko, 1994. Using Visualization to Foster OO Program Understanding, Georgia Tec., TR GIT-GVU-94-33.
9. Myers, B., 1990. Taxonomies of Visual Programming and Program Visualization. J. Visual Languages and Computing, 1: 97-123.
10. Price, B., R. Baecker and I. Small, 1992. A Principled Taxonomy of Software Visualization. J. Visual Languages and Computing, 4: 211-266.
11. Domingue, J., 1998. Visualizing KBS, Chapter 16 of Software Visualization: Programming as a Multimedia Experience by John Stasko, Blaine A. Price and Marc H. Brown (Edr), MIT Press.
12. Ramadhan, H., 2001. Incorporating Software Visualization in the Design of Intelligent Diagnosis Systems for User Programming. J. Artificial Intelligence Review, 16: 1-22.
13. Mackinlay, J., 1986. Automating the Design of Graphical Presentationof of Relational Informaiton. ACM Transactions on Graphics, 5: 110-141.
14. Kuntz, M. and R. Melchert, 1990. Ergonomics Schema Design and Browsing with more Semantics in the Pasta-3 Interface for E-R DBMSs, F. Lochovsky, Ed., North Holland.
15. Aiken, 1998. The Tioga-2 Database Visualization Environment. University of California, Berkeley, TR.