Usability, Databases, and HCI

Fei Li and H. V. Jagadish
Univ. of Michigan

Abstract

As usability is becoming recognized as a crucial feature for databases, there is growing reason for us to pay attention to the principles of Human Computer Interaction (HCI). This article explores the interaction of HCI with databases. We can learn from specific HCI areas, such as information visualization, from general HCI principles, such as direct manipulation, and from HCI methodology, such as running good user studies. However, there is much required for database usability that goes beyond HCI.

1 Usability in Databases

Computer professionals have long recognized the importance of organizing data. The core principles of database technology were established early in the history of modern computing. Today, databases are widely used in a broad range of organizations, and are the basis for a vibrant commercial sector.

In spite of these successes, there has been a longstanding question asked by many about why so much data is not in databases. In comparison, consider compilers, another computing technology whose core principles were developed early. There do remain today a few specialized situations in which people choose to write assembler directly and forgo the benefits of compiling a high level language, but these situations are rare. Why do so many choose so frequently to give up the many wonderful benefits provided by databases?

Over the years, attempts have been made to address this question, most notably through object oriented databases. However, the question has not been a central concern of the database field: when there is so much growth and such great need, it is hard to devote attention to the audiences not being served well.

The web has changed the situation dramatically. If you ask a non-technical person today, they will think of the web as the greatest (distributed) information store and of a search engine (like Google) as the way to (organize and) access this information. Structured databases come only much lower down in perceived importance. Furthermore, the web has democratized information, through disintermediation. When travel agents were the only users of a reservation system, they could be trained to learn magic letter codes and unintuitive access paths. But today, most of us make our own travel arrangements, without a human travel agent as an intermediary. The general user population is not willing to learn magic incantations and gets frustrated with unintuitive access paths. This democratization of database access has led to a greatly increased need for usability.

Not surprisingly, usability issues have gained importance in the database community in recent years. Since usability is a central objective of the human computer interaction (HCI) community, it is natural to ask what we can learn from them. This paper examines some HCI accomplishments that relate to databases, and argues that there still remains much that we need to do beyond that.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

2 Data Management, HCI and User Studies

Computers are notoriously logical: they do precisely what they are instructed to do – no more and no less. Since this is not the way that humans interact with one another, and since what computers do can be quite complex, it is not surprising that many humans find it challenging to interact with computers. HCI developed as a field to address this challenge. The focus in HCI has been on how a human can interact with a computing system to get a task done. Since most tasks that a computer can help with are computational in nature, the traditional focus has been on the computation rather than the data. Only recently has there been enough, and complex enough, digital data that information search has itself become an important computational task. In consequence, database systems have become a more important application domain for HCI work.

When evaluating human interaction with computers, the humans in the loop are central – it is usually not possible to evaluate an HCI system without humans! Therefore, user studies, of many different types, have been used widely by HCI researchers [15]. In contrast, most database systems work is concerned with functionality and performance, which can usually be measured directly from the system, without including the user. As a result, user studies have been rare in database systems work. However, this is beginning to change – there is an increasing number of papers published in database venues that now include a user study. As we in the database field start running more user studies, there is much we can learn from others who have run such studies before, and HCI experts in particular. In other words, in addition to the specific research topics that we will discuss below, there is methodological expertise in the HCI community that is likely to be of value to many in the database community. So it worth our effort to learn from them.

The most common type of user study one sees in a database usability paper has the users performing some task using two (or more) alternative systems, say one (or more) baseline and the new system [50, 37, 36]. For each, the study typically measures time to task completion, and may possibly also measure quality of task performance (such as correctness of response). Care has to be taken to balance the two groups, and this includes considerations that computer systems do not often have, such as learning effects and fatigue effects. Furthermore, user studies are considerably more expensive to run than computer system performance studies, so what is evaluated has to be planned with care. In HCI, and in other fields that routinely run user studies, such as many social sciences, there is considerable attention paid to exactly what the users are told, since user knowledge and expectations can bias observed results [15]. In database user studies, such niceties are not often considered. Similarly, in HCI, one may perform an observational study – just recording where the user spent time and where they were confused. There may also be carefully designed (subjective) questionnaires. While some database user studies do look at such points, there is scope to do so in a much more sophisticated manner.

A new methodological wrinkle is due to remotely run user studies on sites such as Amazon's Mechanical Turk [2]. Some "turkers" may be trying to maximize income as a function of time spent, and so may not put in effort to answer questions carefully – such"cheating" is much less likely when the facilitator is physically present as in a traditional study. Many a researcher has been surprised by nonsense study results due to lack of subject effort. For example, a "turker" may choose the first option in each multiple-choice question, just to move on as fast as possible, possibly without even reading the question. A study designer, in response, may build in a small minimum delay before the next question will load, forcing the "turker" to slow down. The "turker" in turn may counter this by opening multiple windows and performing multiple tasks in parallel, still paying little attention to each, but satisfying the minimum time requirement. The design of studies in such an environment is a small sub-field in itself [25], not really core to traditional HCI, but closely related. It is also believed [7] that for many "turkers," even though the intention is not to cheat overtly, if the instructions are not clear or the question asked is confusing, a common path taken is to guess (or choose randomly) and move on, rather than to put in the effort to clarify. Therefore, the usability of the study itself becomes an important requirement for its success.

Of course, user evaluation is only one piece of most HCI work. The bulk of the contribution from HCI is in designing new tools and techniques that improve usability.

3 Understanding and Interpreting Data

Databases contain structured information, and the results produced can be voluminous. It has long been recognized that humans can benefit from better data presentation, particularly when the data to present are voluminous. Edward Tufte's book [47] is a classic. But this work, important though it is, hardly represents the beginning: rather information visualization has been studied for as long as there has been information, and certainly well before the age of computers. The *diagrammatic* map of the London underground was devised by Harry Beck in 1931 [3]. By not remaining faithful to the actual geography, a much more "readable" map of the subway system was developed than had been there before. Such diagrammatic maps are now typical in most subway systems worldwide. Even older than this is Charles Minard's 1869 chart showing the number of men in Napoleon's 1812 march to Russia [1]. This chart has been widely acclaimed for how well it represents multiple variables, including the size of army, the passage of time, and the geography, all in a single compelling graphic.

3.1 Result Presentation

The typical goal for information visualization is "giving to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space". As researchers have thought about how to present large query result sets, they have developed many innovative ways to look at information.

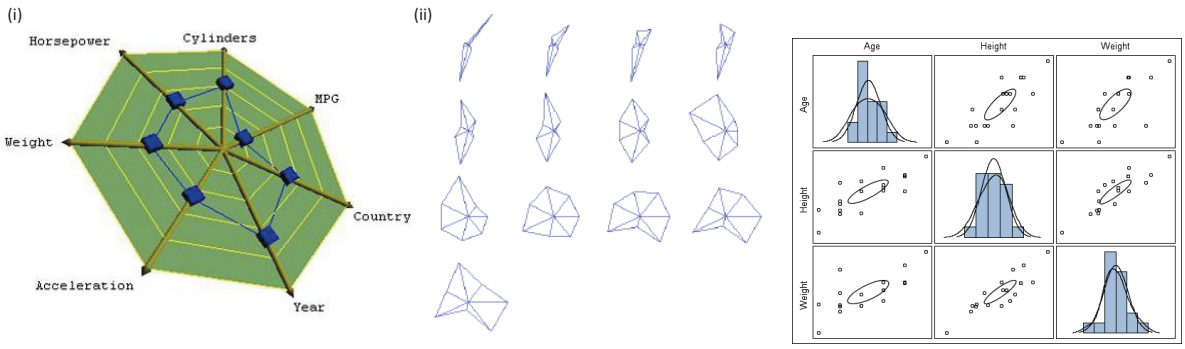
For example, in Star plot [11], each (multi-attribute) data item is shown as a star and all the attributes are represented as equal angular axes radiating from the center of a circle, with an outer line connecting the data value points on each axis. As a result, Star plot is able to clearly present thousands of data items with tens of attributes to the users (Figure 1(a)). Another technique, Scatterplot matrices [24], is designed to show the relationship between different attributes. It enumerates all pairs of attributes, takes each pair of attributes as a projection, maps each projection in one 2D scatterplot, and finally organizes these scatterplots in a scatterplots matrix. By doing this, even if there are many data items in the database, users can clearly observe the correlation between each pair of attributes (Figure 1(b)).

Instead of just listing all data item on screen, some techniques convey the relationship between data items to the users. An example is Narcissus [22], in which the system takes each web page as a node and each hyperlink as an edge, and shows the graph structure of the web (Figure 1(c)).

When displaying these hierarchical (or graph) data, lower levels can quickly get crowded since the number of nodes per level can grow exponentially in a tree (or a graph). To deal with this crowding problem, hyperbolic tree [31] employs hyperbolic space, which intrinsically has "more room" than Euclidean space and clearly visualizes large, wide hierarchical data (Figure 1(d)). Similarly, treemap [42] hierarchically partitions the screen into nested regions and potentially increase the space for showing hierarchical data. Moreover, treemap can also give a good overview of large datasets with multiple ordinal attributes by using a region size to denote an attribute value (Figure 1(e)). Another way to show large graph data is to use interactive distortion techniques, to emphasize part of the data or just to make the space larger. "Overview first, zoom and filter, then details on demand [43]" are the typical processes. For instance, Fisheye Views [18] dynamically show areas of interest larger and with more detail (Figure 1(f)).

Even where the information being searched is not particularly structured, there may be value in presenting the results in a structured way. Search engines group together results from similar sites. Research papers have considered clustering search results based on topic [38]. The crudest examples are for disambiguating homonyms, such as jaguar the animal, the car make, and the football team. But even when there is no such gross disambiguation involved, say given a search for the company "Samsung", we may find it useful to cluster together pages about Samsung products and separately cluster together pages about the Samsung company financials and news mentions.

One way to impose visual structure that has become very popular of late is the *tag cloud* (also sometimes called a *word cloud*). The idea is to choose the most commonly occurring terms (words/tags/...) in a corpus

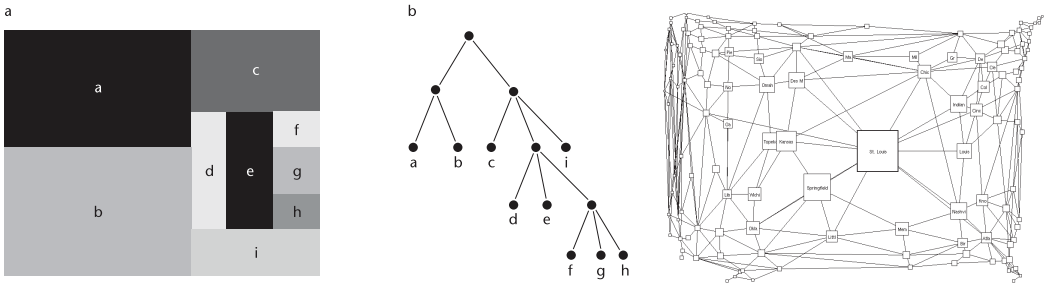


(a) Star plot: (L) The left figure in a is one star glyph, representing seven attributes of a car in which the length in each angular axe shows the value in the corresponding attribute. (R) The right figure shows a group of star glyphs, which take a small space but can potentially show large numbers of star glyphs [12].

(b) Scatterplot matrices: This scatterplot matrix for 3-dimensional data show the relationships between each pair from attribute set {age, weight, and height}.

(c) Narcissus: Representation of complex web structure of a collection of web pages in Narcissus, in which each node is a webpage, each link a reference relationship. The large nodes are indexes into the pages, and the ball-like structure represents the set of cross referenced pages [22].

(d) Hyperbolic tree: This Hyperbolic tree locates its root in the center of the space and all its offsprings in the hyperbolic plane hierarchically, and clearly shows the layout of a tree with 1004 nodes [31].



(e) Treemap: The TreeMap in the left show the hierarchical structure of the tree structured data in the right [42].

(f) Fisheye Views: Fish-eye view of links between major American cities with a focus on St. Louis [39].

Figure 1: Examples in Information Visualization Techniques.

and to show these in a manner that informs the user. Typically, the font size is used to convey frequency of occurrence. The boldness and color of font, as well as placement of words in the cloud, can be used to represent information about the unstructured corpus in a succinct way.

See [10, 41] for a good collection of research works on information visualization. The ManyEyes system [49, 48] has implemented many of the better known information visualization methods.

3.2 Visual Analytics

Crudely speaking, database applications are divided into two types: the first is operational or transactional, and the second is warehousing. Typically, the former do not produce large or complex results. In consequence, much of the focus of information visualization has been on the latter. But the typical consumer of information from a data warehouse is a decision-maker, who is using the displayed warehouse information to better understand and analyze it in support of the decision to be made. It is also well-known that a typical decision-maker does not issue a few queries in one shot, get results and base all decisions on the results of these queries. Rather, there is a sequence of queries, examining data that is surprising from multiple angles, drilling down into outlier aggregates, running what-if analyses, and so on. In short, we have a user task accomplished through a sequence of queries, each of which produces a query result that can be visualized using suitable techniques such as those mentioned in the preceding sub-section. This bigger picture then begs the question why the user has to look at well-presented information and then develop a query in a separate window/interface. Couldn't we short-circuit this, have the user directly interact with the visualized information, and thereby analyze data from the warehouse more effectively? The HCI principle of direct manipulation certainly suggests this

The epiphany above gave rise to the field of *visual analytics* [45], also sometimes referred to as *visual data mining*. This is also central to the commercially sold Tableau system, based on VizQL [21]. But let's step back for a moment from the picture painted in the preceding paragraph, and think about what the user is really trying to do. The user wishes to find interesting patterns or items worthy of attention, and is doing so through a process that involves a repeated cycle of visualization and interactive commands. In other words, the human and the computer system are symbiotically solving a problem: the human is looking for patterns in the visually presented data and also thinking about what presentations may be most informative, while the system is crunching through large quantities of data to develop informative visual presentations. Developing such a system requires the developer to think about how to make the human most effective at finding patterns of interest. In contrast, the traditional data mining problem is to develop a system that can itself find patterns of interest. Where the pattern of interest is easy to define precisely, there is no question that the system can do very well. But where, as is often the case, there could be many patterns of interest or patterns are hard to define, then the flexibility of the human brain is a tremendous advantage, and one should expect visual analytics to be far more powerful than traditional computer-driven analytics.

4 Query Specification

The preceding discussion primarily dealt with the presentation of information from a database, and indeed this is where the bulk of the work is, in dealing with information sources. However, before we get to look at the answer, we have to pose the question and get the expected query results. In a traditional database system, this is not easy. Standard query languages, like SQL or XQuery, while expressive and powerful, require the users to learn both the database schema and how to compose the exact query statements. Since this learning curve is often too steep for naive users, usable query mechanisms are in dire need. Ideally, a perfect usable query system should be like a private intermediary agent who is a database expert and works tirelessly for the end-users. The end-user thinks of what she wants to query and expresses this thought in her own idiosyncratic way to the intermediary agent. Because of the knowledge gap, the intermediary agent needs to figure out what the user exactly wants to query.

In the old days, this intermediary agent may have been a staffer in a computer support group. Today, the delay of going through such a staffer is usually unacceptable, even if the cost were acceptable. Therefore, our desire is to build a computer system that can perform this role of intermediary agent. In this section, for each query mechanism, we first describe some state-of-the-art works in the intermediary agent perspective. Then we discuss the gap between these state-of-the-art works and the ideal usable system in the same query mechanism, and finally try to figure out possible ways to improve these existing works.

4.1 Visual Interfaces

Forms are the traditional way for naive users to query a database. The designer of the interface carefully designs one or more forms by making an educated guess of which data the users would be interested in and how these desires may be mapped to query statements in the database. Then she gives these forms to the intermediary agent, who converts the form into a DBMS query. Forms work very well when the query logic of the end-users can be expressed by these forms. But, what if the end-user wants a query that cannot be expressed by the existing forms? Since the designer cannot design all forms for all the end-users, she could give some form generation rules to the intermediary agent [27, 28], letting the intermediary agent generate more forms and hoping some of these forms can express the expected query logic. The intermediary agent may even be able to take keywords as a preliminary input, gives the forms related to these keywords to the end-users [13], and hopes one of these forms can exactly express the expected query logic. The ideal form-based agent should always be able to provide one form which can express the expected query logic of the end-users. That is, the user tells the agent what information she needs in her own way and the agent generates the form on the fly. Getting closer to this ideal remains a current topic of research.

There are also many endeavors in non-form visual query mechanisms that enable end-users to compose natural, generic query logics. These works began in 1975 with QBE [51], which allows users to query a database by creating example tables in the query interface. The agent then translates them into standard query language, and finally executes it on the database. After QBE, many other visual query languages have been proposed in academia, like QBT [40], XQBE [9], MIX [34], Xing [16], Kaleidoquery [35]. In industry also, visual query specification (and visual database design) are commonly supported. For example, Microsoft Access supports QBE-based query over relational databases and IBM uses Visual XQuery Builder to support visual queries over XML databases. While expressive, these methods require the users to specify how these data are structured and where to look for the information they desire, which makes them unsuitable for totally naive users. An ideal agent should allow the user to input the examples in the user's paradigm, not in the system's paradigm, then analyze what the users want to query by these examples (with some feedback for confirmation if needed), and finally translate these examples into standard query statements.

4.2 Text Interfaces

Natural language interfaces for databases (NLIDBs) have a long history [6]. Most NLIDB systems, like Lunar [17], parse the query into a parse tree and then map the parse tree into a database query expression. Also, some interactive NLIDB query systems [29, 46] are proposed to guide the query formulation interactively and adaptively. However, most of these early works suffer from both domain dependency and database dependency, which rely heavily on domain knowledge and manually created adaptation rules for each database.

Today, the most widely used query mechanism for naive users is the keyword-based query interface. Without the requirements of any schema information or any programming skills, the agent only asks the users for some keywords and then returns the data that are relevant to these keywords. In general, the relevant data is a group of closely inter-connected tuples, according to foreign-primary key relationship (in relational databases [8, 5, 23]) or parent-children relationship (in XML databases [14, 20]), that contain all the keywords.

However, many user needs cannot be satisfied with just any relevant data provided by the agent. . Unfortunately, a bag of keywords often does not capture enough information about the user's information need, and the system's guess at completing this need is either wrong, or too non-specific [26].

To make the search more expressive and effective, some recent works try to figure out the semantic information in the keywords (e.g. SQAK [44]) or enable the users to specify some semantic information in addition to keywords (e.g., Schema-Free XQuery [33], natural language [32], query based on schema summary [50]). SQAK [44] extracts the aggregation information from flat structured keywords. But this work falls far short of removing all the ambiguity in keywords. Schema-free XQuery [33] integrates keyword search functionality into XQuery as built-in functions and enables users to query XML documents based on partial knowledge they may have about the database schema. Furthermore, NaLIX [32] provides a generic natural language query interface, instead of keywords, to XML database. Since natural language itself can convey semantic information, it reduces the uncertainty resulting from the intrinsic ambiguity of flat keywords. Another work, query schema summary [50], generates schema summary and supports queries on this simple schema summary rather than on the original complex schema.

5 Conclusion

As the use of databases is "democratized," in that end-users interact with them directly rather than being mediated by MIS staff, the importance of usability in databases is growing. There is much that we as a community could do in this regard, and much that we can learn from HCI. However, HCI alone is not going to solve all our usability problems: there is now a bonafide sub-field of database usability, which squarely deals with databases while drawing inspiration from work in HCI. This article provides an overview of the issues.

Since this article is written for a database audience, the bulk of it is devoted to the impact of HCI on databases. This does not imply that there isn't impact in the other direction. It is now possible to record every action performed by the user, and the time taken to perform these actions. With a little bit of instrumentation, one can go even further, to track eyeball movements, for example. This gives rise to a significant size database from a single user session. Analyzing patterns in this database has already been demonstrated to be of value in developing better interfaces in a range of applications from web search engines [19] to video games [30]. In short, there is much that database technology can contribute to HCI, just as in the reverse direction.

6 Acknowledgment

This work was supported in part by NSF under grants IIS 0741620 and IIS 1017296.

References

[1] All graphic works of charles joseph minard: http://www.datavis.ca/gallery/minbib.php.

[2] Amazon mechanical turk: http://aws.amazon.com/mturk/.

[3] London subway: http://diagrams.org/fig-pages/f00022.html.

[4] Visualizing higher dimensional data: http://www.mathworks.com/products/demos/statistics/mvplotdemo.html.

[5] S. Agrawal, S. Chaudhuri, and G. Das. Dbxplorer: A system for keyword-based search over relational databases. In *ICDE*, pages 5–16, 2002.

[6] L. Androutsopoulos. Natural language interfaces to databases - an introduction. *Journal of Natural Language Engineering*, 1:29–81, 1995.

[7] M. Bernstein. Personal communication, 2012.

[8] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using banks. In *ICDE*, pages 431–440, 2002.

[9] D. Braga, A. Campi, and S. Ceri. Xqbe (xquery by example): A visual interface to the standard xml query language. *ACM Trans. Database Syst.*, 30(2):398–443, June 2005.

[10] S. K. Card, J. D. Mackinlay, and B. Shneiderman, editors. *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[11] J. Chambers, W. Cleveland, B. Kleiner, and P. Tukey. Graphical Methods for Data Analysis. *The Wadsworth Statistics/Probability Series. Boston, MA: Duxury*, 1983.

[12] W.-Y. Chan. A survey on multivariate data visualization. *Technical Report*, 2006, HKUST.

[13] E. Chu, A. Baid, X. Chai, A. Doan, and J. Naughton. Combining keyword search and forms for ad hoc querying of databases. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, SIGMOD '09, pages 349–360, New York, NY, USA, 2009. ACM.

[14] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. Xsearch: A semantic search engine for xml. In *VLDB*, pages 45–56, 2003.

[15] J. F. Dumas and J. C. Redish. *A Practical Guide to Usability Testing*. Greenwood Publishing Group Inc., Westport, CT, USA, 1993.

[16] M. Erwig. A visual language for xml. In *Proceedings of the 2000 IEEE International Symposium on Visual Languages (VL'00)*, VL '00, pages 47–, Washington, DC, USA, 2000. IEEE Computer Society.

[17] W. W. et al. *The Lunar Sciences Natural Language Information System: Final Report.* Bolt Beranek and Newman Inc., Cambridge, MA, 1972.

[18] G. W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '86, pages 16–23, New York, NY, USA, 1986. ACM.

[19] Z. Guan and E. Cutrell. An eye tracking study of the effect of target rank on web search. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 417–420, New York, NY, USA, 2007. ACM.

[20] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. Xrank: Ranked keyword search over xml documents. In *SIGMOD Conference*, pages 16–27, 2003.

[21] P. Hanrahan. Vizql: a language for query, analysis and visualization. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 721–721, New York, NY, USA, 2006. ACM.

[22] R. J. Hendley, N. S. Drew, A. M. Wood, and R. Beale. Case study: Narcissus: visualising information. In *Proceedings of the 1995 IEEE Symposium on Information Visualization*, INFOVIS '95, pages 90–, Washington, DC, USA, 1995. IEEE Computer Society.

[23] V. Hristidis and Y. Papakonstantinou. Discover: Keyword search in relational databases. In *VLDB*, pages 670–681, 2002.

[24] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91, 1985.

[25] P. G. Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS*, 17(2):16–21, Dec. 2010.

[26] H. V. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu. Making database systems usable. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 13–24, New York, NY, USA, 2007. ACM.

[27] M. Jayapandian and H. V. Jagadish. Automating the design and construction of query forms. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 125–, Washington, DC, USA, 2006. IEEE Computer Society.

[28] M. Jayapandian and H. V. Jagadish. Automated creation of a forms-based database query interface. *Proc. VLDB Endow.*, 1(1):695–709, Aug. 2008.

[29] E. Kapetanios and P. Groenewoud. Query construction through meaningful suggestions of terms. In *Proceedings of the 5th International Conference on Flexible Query Answering Systems*, FQAS '02, pages 226–239, London, UK, UK, 2002. Springer-Verlag.

[30] J. H. Kim, D. V. Gunn, E. Schuh, B. Phillips, R. J. Pagulayan, and D. Wixon. Tracking real-time user experience (true): a comprehensive instrumentation solution for complex systems. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 443–452, New York, NY, USA, 2008. ACM.

[31] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In I. R. Katz, R. L. Mack, L. Marks, M. B. Rosson, and J. Nielsen, editors, *CHI*, pages 401–408. ACM/Addison-Wesley, 1995.

[32] Y. Li, H. Yang, and H. V. Jagadish. Nalix: an interactive natural language interface for querying xml. In *SIGMOD Conference*, pages 900–902, 2005.

[33] Y. Li, C. Yu, and H. V. Jagadish. Schema-free xquery. In *VLDB*, pages 72–83, 2004.

[34] P. Mukhopadhyay and Y. Papakonstantinou. Mixing querying and navigation in mix. In R. Agrawal and K. R. Dittrich, editors, *ICDE*, pages 245–254. IEEE Computer Society, 2002.

[35] N. Murray, N. Paton, and C. Goble. Kaleidoquery: a visual query language for object databases. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '98, pages 247–257, New York, NY, USA, 1998. ACM.

[36] L. Qian, M. J. Cafarella, and H. V. Jagadish. Sample-driven schema mapping. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 73–84, New York, NY, USA, 2012. ACM.

[37] L. Qian, K. LeFevre, and H. V. Jagadish. Crius: user-friendly database design. *Proc. VLDB Endow.*, 4(2):81–92, Nov. 2010.

[38] D. Ramage, P. Heymann, C. D. Manning, and H. Garcia-Molina. Clustering the tagged web. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 54–63, New York, NY, USA, 2009. ACM.

[39] M. Sarkar and M. H. Brown. Graphical fisheye views. *Commun. ACM*, 37(12):73–83, Dec. 1994.

[40] A. Sengupta and A. Dillon. Query by templates: a generalized approach for visual query formulation for text dominated databases. In *Proceedings of the IEEE international forum on Research and technology advances in digital libraries*, IEEE ADL '97, pages 36–47, Washington, DC, USA, 1997. IEEE Computer Society.

[41] B. Shneiderman and B. B. Bederson. *The Craft of Information Visualization: Readings and Reflections*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[42] B. Shneiderman and C. Plaisant. *Treemaps for space-constrained visualization of hierarchies*. http://www.cs.umd.edu/hcil/treemap-history/index.shtml, 1998-2009.

[43] B. Shneiderman and C. Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition)*. Pearson Addison Wesley, 2004.

[44] S. Tata and G. M. Lohman. Sqak: doing more with keywords. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 889–902, New York, NY, USA, 2008. ACM.

[45] J. J. Thomas and K. A. Cook. A visual analytics agenda. *IEEE Comput. Graph. Appl.*, 26(1):10–13, Jan. 2006.

[46] A. Trigoni. Interactive query formulation in semistructured databases. In *Proceedings of the 5th International Conference on Flexible Query Answering Systems*, FQAS '02, pages 356–369, London, UK, UK, 2002. Springer-Verlag.

[47] E. R. Tufte. *The visual display of quantitative information*. Graphics Press, Cheshire, CT, USA, 1986.

[48] F. B. Viégas and M. Wattenberg. Shakespeare, god, and lonely hearts: transforming data access with many eyes. In *JCDL*, pages 145–146, 2008.

[49] F. B. Viégas, M. Wattenberg, F. van Ham, J. Kriss, and M. M. McKeon. Manyeyes: a site for visualization at internet scale. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1121–1128, 2007.

[50] C. Yu and H. V. Jagadish. Querying complex structured databases. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 1010–1021. VLDB Endowment, 2007.

[51] M. M. Zloof. Query-by-example: the invocation and definition of tables and forms. In *Proceedings of the 1st International Conference on Very Large Data Bases*, VLDB '75, pages 1–24, New York, NY, USA, 1975. ACM.