User-Centered Data Management
Synthesis Lectures on Data Management

Editor
M. Tamer Özsu, University of Waterloo

Synthesis Lectures on Data Management is edited by Tamer Özsu of the University of Waterloo. The series will publish 50- to 125 page publications on topics pertaining to data management. The scope will largely follow the purview of premier information and computer science conferences, such as ACM SIGMOD, VLDB, ICDE, PODS, ICDT, and ACM KDD. Potential topics include, but are limited to: query languages, database system architectures, transaction management, data warehousing, XML and databases, data stream systems, wide scale data distribution, multimedia data management, data mining, and related subjects.

User-Centered Data Management
Tiziana Catarci, Alan Dix, Stephen Kimani, and Giuseppe Santucci
2010

Data Stream Management
Lukasz Golab and M. Tamer Özsu
2010

Access Control in Data Management Systems
Elena Ferrari
2010

An Introduction to Duplicate Detection
Felix Naumann and Melanie Herschel
2010

Privacy-Preserving Data Publishing: An Overview
Raymond Chi-Wing Wong and Ada Wai-Chee Fu
2010

Keyword Search in Databases
Jeffrey Xu Yu, Lu Qin, and Lijun Chang
2009
User-Centered Data Management

Tiziana Catarci
University of Rome, “La Sapienza,” Italy

Alan Dix
Lancaster University, UK

Stephen Kimani
CSIRO ICT Centre, Australia

Giuseppe Santucci
University of Rome, “La Sapienza,” Italy

SYNTHESIS LECTURES ON DATA MANAGEMENT #6
ABSTRACT
This lecture covers several core issues in user-centered data management, including how to design usable interfaces that suitably support database tasks, and relevant approaches to visual querying, information visualization, and visual data mining. Novel interaction paradigms, e.g., mobile and interfaces that go beyond the visual dimension, are also discussed.

KEYWORDS
usability, user-centered design, visual data access, information visualization, visual data mining, universal usability
## Contents

Preface .............................................................................................................. ix

1 Why User-Centered .......................................................................................... 1
  1.1 An Example – What Can Go Wrong? ......................................................... 1
  1.2 The Process of User-Centered Design ...................................................... 3
  1.3 Usable Systems ........................................................................................... 4
  1.4 Cost-Benefits of Usability .......................................................................... 5

2 The Early Days: Visual Query Systems ......................................................... 9
  2.1 Visual Representation .................................................................................. 10
  2.2 Visual Interaction ....................................................................................... 15
  2.3 Visual Perception ....................................................................................... 18

3 Beyond Querying ............................................................................................. 19
  3.1 Information Visualization .......................................................................... 19
      3.1.1 Data Extraction and Elaboration ....................................................... 21
      3.1.2 Data Representation ......................................................................... 24
      3.1.3 Data Presentation ........................................................................... 33
  3.2 Visual Data Mining ...................................................................................... 37
      3.2.1 Background ....................................................................................... 40
      3.2.2 Definition and Importance of Visual Data Mining ......................... 41
      3.2.3 Visual Data Mining Approach ......................................................... 42
      3.2.4 Relevant Systems ............................................................................ 43
      3.2.5 Challenges and Issues ..................................................................... 47

4 More Advanced Applications .......................................................................... 49
  4.1 Web Data ................................................................................................... 49
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.1</td>
<td>The Changing Face of the Web</td>
<td>49</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Understanding Web Data</td>
<td>50</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Examples of End-User Web Data Interfaces</td>
<td>53</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Web Technology and the Web of Data</td>
<td>54</td>
</tr>
<tr>
<td>4.1.5</td>
<td>Interacting with the Web of Data</td>
<td>59</td>
</tr>
<tr>
<td>4.1.6</td>
<td>Web Data or All Data?</td>
<td>69</td>
</tr>
<tr>
<td>4.2</td>
<td>Mobile Interfaces</td>
<td>69</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Challenges and Opportunities of Mobile Computing</td>
<td>70</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Usability and Mobile Computing</td>
<td>71</td>
</tr>
<tr>
<td>5</td>
<td>Non-Visual Interfaces</td>
<td>75</td>
</tr>
<tr>
<td>5.1</td>
<td>Accessibility</td>
<td>75</td>
</tr>
<tr>
<td>5.2</td>
<td>Aural Interfaces</td>
<td>77</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions</td>
<td>83</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>85</td>
</tr>
<tr>
<td>Authors' Biographies</td>
<td></td>
<td>95</td>
</tr>
</tbody>
</table>
This lecture was initially intended to cover relevant issues in database user interfaces, mainly query interfaces. However, very soon, we realized that providing friendly access to information is much more than just designing nice interfaces; rather, it has to do with designing interactive systems that suitably fit the users’ tasks, and this can be achieved by following a user-centered approach to system design. Second, nowadays the data that users want to access do not reside only in traditional databases, they are mainly on the web (available or hidden – it does not matter). Third, users do not limit themselves to just extracting the data; instead, they want to manipulate them, analyze them, and make sense out of them. Thus, user-oriented systems should provide more functionality, in addition to querying. Finally, while visual interfaces and information visualization techniques are usually considered the most usable approaches, categories of users and/or contexts exist for which they are not appropriate, so other interactive paradigms need to be explored.

The content of this lecture derives from all the above considerations. Indeed, the lecture starts in Chapter 1 by discussing the importance of adopting a user-centered approach. Chapter 2 takes the reader to the early days, where we find the initial use of visual interfaces to support database tasks. Visual representation, interaction, and perception are discussed. In Chapter 3 the discussion moves on to shed more light on two concepts behind database querying. The focus here is on information visualization and visual data mining. The discussion, in Chapter 4, then describes non-traditional interfaces that are relevant to databases. In particular, the chapter looks at web data and mobile interfaces. Chapter 5 describes interfaces that go beyond the visual dimension. In particular, it discusses accessibility and aural interfaces.

Tiziana Catarci, Alan Dix, Stephen Kimani, and Giuseppe Santucci
June 2010
CHAPTER 1

Why User-Centered

In data management, the needs of the underlying technology often appear most demanding: choosing appropriate data formats, schema, and perhaps, importing data from heterogeneous sources into consistent relational databases. At best, the user interface is an afterthought, and, at worst, users can be seen as an inconvenience, people who may enter incorrect or inconsistent data and, therefore, should be protected against.

It is also obvious that the data are really ultimately there for people to use or benefit from; in other words, it is the users not the data that are important. However, it is easy for this ultimate purpose of data to be overshadowed by the more immediate and unforgiving constraints of technology.

A user-centered approach puts people at the heart of design. This is not just about the aesthetics of the screens “putting a nice user interface on it,” but it permeates every aspect of the design of a system. If even the lowest-level of data is structured badly, it is very hard to create usable systems on top of it, and ‘bad’ here does not mean in terms of some sort of internal consistency, but it is, more fundamentally about being fit for purpose or fit for ultimate use.

This focus on the users is at the heart of user-centered design, in general. To some extent, user-centered data management is simply an application of this focus on the users and can incorporate general techniques that can be found in standard textbooks in the area (Dix et al., 2004; Preece et al., 1994; Shneiderman et al., 2009) and international standards, principally ISO 9241 and ISO 13407.

The rest of this first chapter gives a short introduction to some of these general principles. Section 1.1 gives a short motivating example that demonstrate some of the pitfalls when systems are designed without a user-centered focus. Section 1.2 then discusses the process of applying user-centered design and Section 1.3 the general notion of ‘usability.’ Finally, Section 1.4 looks at the cost–benefit trade-off when applying user-centered design, how to decide when you have done enough work.

Although these general principles of user-centered design can be applied on a bespoke basis to particular applications, there are also special issues that arise when we consider the users of applications in data management and particular kinds of user interfaces. It is on this more specific area of user-centered data management that we focus in the remainder of this lecture.

1.1 AN EXAMPLE – WHAT CAN GO WRONG?

To see why this focus on the user is so important, let’s consider an example the authors encounter repeatedly. We are academics on examination boards when student grades are being discussed. It is common to see a departmental spreadsheet or bespoke database being used as the primary reference,
2 1. WHY USER-CENTERED

not the official university data system. Later, the university system is updated from the department one, but it is the local data that are deemed canonical, not the ‘official’ central one. Similar stories can be told in industrial and governmental settings.

What is going on in such cases?

Sometimes, this is a matter of local politics where individuals want to retain a sense of control. One could argue that they should not do this, but, of course, this is natural, and any information system design that ignores these personal and political realities will fail.

Sometimes, it is due to poor usability of the central university information system. For example, most of the time during the exam board some sort of summary list of grades is used. However, if a grade needs to be altered it is often not possible to edit this in the list view, but, instead, one has to drill into the student’s individual record or even navigate out of the list view and into a single student view. This is, perhaps, the easiest kind of problem to fix with ‘add on’ solutions: new screens or different means to update.

Sometimes, there is an intrinsic problem with the data structures. For example, it is common to attempt to enter the marks for a course, only to find that the central system refuses to allow the marks of one of the students to be entered. This is because, for some reason, the student is not formally registered for the course on the system and the marks table in the database requires the registration record as an integrity condition. While this makes perfect sense when looking at the internal database or low level business logic, it then causes problems under real use when the world is not perfectly consistent. While this can be corrected, the paperwork often takes some time, during which the student’s marks cannot be collated, and the mark-sheets that were being used to enter the data may even be misplaced.

Note that in this last case, the fundamental data structures have favored consistency over currency and, in so doing, work against the human systems within which they need to work. This is not a case where a user interface can be ‘bolted on’ to an existing data system; the human social and organizational needs must be considered at the data design stage. In real use, it is common for there to be ‘inconsistent’ cases, that is, for the data that arises in real life to, in some way, break rules or constraints that would hold of ‘ideal’ data. This often happens temporarily during updates, but it can be longer lasting when information in the system is partial or incomplete (the student is de facto taking the course; he/she is just, for some reason, not registered in the system). However, data design often focuses on ensuring a consistent internal representation, however, inconsistent with reality. Often, it is better to adopt a data design that allows inconsistency, but it is, in some way, highlighted or flagged as needing attention: let the marks be entered, and raise a ‘to do’ that indicates something needs to be fixed.

In some cases, user needs can be incorporated in generic data design heuristics (such as allowing but highlighting inconsistency above). In many cases, one needs a much more detailed understanding of the particular nature of the users of the system and the context in which it will be used.
1.2 THE PROCESS OF USER-CENTERED DESIGN

There are almost as many ways to pursue a user-centered design approach as there are practitioners. However, the process in ISO 13407 human-centered design process is a good place to start as it is best based on practice and since publication heavily influences practice.

ISI 13407 identifies four principle activities:

(a) *Understand and specify the context of use* – Who are the potential users? What is the environment in which the system will be used?

(b) Understand and specify the user and business requirements – What do we want the proposed system to do?

(c) *Design the product* – Create a design/prototype that meets the goals identified in (c), given the contextual constraints in (d).

(d) Evaluation of the design – Does the design that has been created in (c) actually work?

Sometimes (a) and (b) are lumped together under a general heading of ‘understanding’ ‘requirements’ or ‘analysis,’ although they differ in that (a) is principally about what will not change once a new system is created, whereas (b) is precisely about what is hoped will change. However, the distinction is always fluid, as Schon, D. (1991) has argued, it is often the most successful designers who question the problem formulation; the context itself may need to be changed.

These first stages will typically involve observing, interviewing, or meeting with real end users, although this can be difficult, especially for widely dispersed users such as ‘any web user’ or very busy users such as senior managers. When users include small children, there can be special difficulties both in understanding their point of view and also because of legal constraints (Markopoulos et al., 2008). However, for all types of users, there are two fundamental problems. The first is trying to understand people who are often very different from yourself as designer. The second is, perhaps, more subtle, which is understanding the things that you share with users; as so much of our knowledge is tacit, done without being aware of it, sometimes, it can be more difficult to understand those closest to you.

Many people suggest using some form of participatory design approach (Greenbaum and Kyng, 1991) where the end users are brought into the design process and invited to become, not just subjects to be studied by the designers, but, effectively, part of the design team or co-designers. Note that this helps to address the first of the problems above, the otherness of users, but the analyst/designer still needs to use expert facilitation in order to uncover users’ tacit knowledge.

There are many methods used in the design phase, but most end in some sort of prototype, although this can vary from sketches of a proposed interface, to be shown to prospective users, to fully working software. The prototype is particularly important in order to evaluate the design with real users.
1. WHY USER-CENTERED

Alternatively, some form of expert evaluation is used where a usability expert uses some forms of checklist or heuristics to guide a structured analytic assessment of the design. This is often performed on a screen-by-screen (or on the web page-by-page) fashion, for example, by asking, whether it is clear what to do next from the current screen.

There are generic heuristics (see Box) which apply to most kinds of systems, but they were originally developed during the creation of desktop systems. However, there are also heuristics for more specific domains such as mobile interfaces (Bertini et al., 2009).

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Nielsen’s Top 10 Usability Heuristics 1–9 from (Nielsen, J., 1994) all 10 in (Nielsen, J., 2005)

Perhaps, the most crucial thing is that these stages of analysis, design, and evaluation occur in a cycle. There are small cycles, for example, during design gap in understanding of the users context or goals may become apparent. Similarly, evaluation may show how the design fails to meet the user requirements. However, there is also a bigger cycle, where the process of evaluation highlights aspects of the context or requirements that were otherwise missed. This is particularly important when creating radically new systems. Users find it hard to envisage something dramatically different from the status quo, but when faced with a potential design, they are often able to articulate potential uses that had previously not been considered.

1.3 USABLE SYSTEMS

Usability is a complex term including everything from surface aesthetics to organizational fit. ISO 9241 defines usability in terms of three factors:

Effectiveness – The accuracy and completeness with which specified users can achieve specified goals in particular environments.

Efficiency – The resources expended in relation to the accuracy and completeness of goals achieved.

Satisfaction – The comfort and acceptability of the work system to its users and other people affected by the work system’s use.

This is a rather dry, committee definition but basically addresses the following:
Effectiveness – Does it do the job?
Efficiency – How easily does it do the job?
Satisfaction – How enjoyable (or at least un-stressful) is it to do the job?

While this can be interpreted to apply at the level of workgroups or organisations, it is more common to look only at the user directly interacting with a system. However, it is important to keep one’s focus more broadly on all stakeholders. The interface to a CRM (customer relationship management) system at a call center does not just affect the telephonist taking the call, but also the customer at the other end, even though the customer does not directly interact with the CRM system or even know that it exists.

For many years, the first two factors in the ISO standard, effectiveness and efficiency, were dominant in usability thinking – basically allowing people to work better. However, gradually, the last of these, namely satisfaction, has become more important for several reasons. First, it has been widely recognized in work situations that a happy worker is a more productive worker! Second, enjoyment may now be the primary objective as computer systems have spread out of the office and into the home and public places. Finally, as many systems are now delivered over the web, applications that were once shrink wrapped (e.g., Word Processing) are now delivered as services (e.g., Google docs); while a product is delivered once only with a single purchase decision, services have many decision points – it is easy to change services and more importantly to maintain the users’ good will. As we shall see in Section 4.1, this shift to web delivery also includes many traditional data services.

This shift of concerns has lead to the notion of user experience as part of, or often subsuming, usability. Instead of seeing users merely as operators, their overall experience of using a product or service becomes a primary goal.

Of course, users are not all the same, and this has led to a focus on customizability and personalization, both for desktop applications and for those delivered over the web. Perhaps, most importantly, we are not all the same in terms of our physical, perceptual, and cognitive abilities. With an aging population, ‘disability’ of some kind or another will become the norm – ignoring those with reduced or different abilities may cut you off from a large segment of the population. However, it is not just good ethics and good for business that one should cater for a wide range of abilities; in many countries, accessibility legislation demands that usability really means usability for everyone. Cutting corners on usability can land you in court. In Section 5.1, we examine some of the implications of accessibility for data management.

1.4 COST-BENEFITS OF USABILITY

Focusing on users takes time, effort, and money; it can be hard to quantify benefits and hence to determine just how much effort is worth spending to make improvements (although one can make good estimates Bias and Mayhew, 2005). Given this, it may be tempting to either ignore it as an issue entirely or, alternatively, to obsessively attempt to iron out every slight difficulty.

However, in order to avoid the high cost of poor utilization and produce a successful system, usability must be made a priority. In the example at the beginning of this section, we saw that
poor usability often means people maintain parallel information systems. Also, if it is difficult to enter pertinent information, they may simply not bother to do so. Even though the problems are with superficial usability, political and social context are fundamental to the design of the data structure; in all cases, when people in an organisation keep parallel systems, the centrally held 'official' organizational data becomes unreliable and inconsistent with reality. Paradoxically, the attempt in the database to maintain data integrity, in the narrow sense of being a self-consistent model, actually works against real integrity so that the data held begin to diverge from the things in the real world they are intended to represent.

However, things can be much worse. In 1992 the London Ambulance installed a new computer-aided dispatch; although it was intended to be a new sleek automated system, it became a potentially fatal disaster (Sommerville, I., 2004). As soon as it was deployed, it became clear that, rather than streamlining dispatch, delays increased, and a short while after deployment, the entire system collapsed leading to an emergency re-introduction of the old manual system. There were numerous failures but not the least amongst these was a failure to take into account the complex human and organizational factors involved in the dispatch process. Although there was some debate, no deaths were unambiguously attributed to the incident, this was only by good fortune. Poor usability can kill.

Happily, the costs of usability failure are not always so severe, and in practice, one must strike a balance between the efforts spent making a system more usable and the benefits of the improvements.

In fact, there are two stages to this. First, find potential problems or potential things to improve; second, decide whether it is worth implementing a solution. For the latter, it is possible, in principle, to do a simple cost–benefit trade-off. Figure 1.1 shows this with the severity of a usability problem on the left axis and the cost to fix on the bottom (focusing here on problems, a similar analysis applies to potential improvements).

![Figure 1.1: Cost-benefit trade-off for usability problems.](image-url)
1.4. COST-BENEFITS OF USABILITY

Easy cases are found at the top left and bottom right. In the first, (a) we have problems that are severe and have an easy and obvious fix, and, of course, one deals with these first. At the bottom right, (d) these are the minor problems that are hard to fix whereas, often, one must simply accept that they are there. An example of this might be a database configuration screen that is hard to understand. If configuration is only done occasionally and those doing it are highly trained, then we may simply assume they will read the documentation thoroughly on each occasion.

The cases at the top right and bottom left are where the decision on what action to take is more difficult even as the cost–benefit is more evenly balanced. On the bottom left (c) are minor problems that are easy to fix. Usually, it is worth dealing with these as they may mask other problems or generally reduce users’ confidence in the system. However, they will, of course, have less priority than those at (a). Finally, at (b), there may be problems that have serious impact but are expected to be hard to fix. These are most problematic as some action is needed, but the costs of a ‘proper’ fix may be too high. In this case, the best option may be to seek workarounds that are not perfect but, in some way, limit or mitigate the problem.

One example of this last case (b) occurred in the online booking system of a large UK hotel chain. Web-based systems often have problems when users press ‘back’ in the middle of an interaction or use history, especially when viewing the confirmation page after completing a web form. In the case of this hotel, the effect was that the user could end up duplicating a booking. There are various ways this can be solved, but they often require a substantial re-engineering of the code. Instead of doing the ‘proper’ fix, they changed the system to simply refuse to create two identical bookings on the same day. This meant that if you did want to book two rooms in the same hotel, you needed to either do it as a single transaction, or wait a day for the second booking. However, this was a rare occurrence; the page that rejected the duplicate booking explained the cause, and it was a lot less damaging than accidentally repeated bookings.

Of course, the picture is somewhat simplistic. Severity is a multi-dimensional concept, which includes the likelihood of these particular problems: the number and kind of users who may encounter it (avoiding giving the CEO headaches) and the impact of the problem (death or minor inconvenience). The label ‘cost to fix’ also assumes one can assess this cost, and as in any area, this is a combination of experience and judgement. However, while these are not metric scales, it is still often not hard to rank issues against both these axes and so get some sense of the best place to focus effort.

Furthermore, this also assumes that we know what needs to be fixed. Finding problems is also a matter of the cost–benefit trade-off.

To find errors in a system, one of the most common ways is to try a prototype out with real users. This can be a costly business, both for the analyst doing the study and for the users who will need to stop doing their normal jobs while taking part in the study. Some years ago Nielsen and Landauer (1993) studied a number of large software projects in order to assess the optimal number of users to study in each iteration of a design. They found that there were diminishing returns in studying extra users, as many problems uncovered by a new user had already been uncovered by earlier users.
1. WHY USER-CENTERED

- the exact proportions varied between projects, but often, they found that 70-80% of errors were uncovered by the first few users. Projects also differed in the costs of a re-design cycle during an iterative development cycle, and as a re-design could create fresh problems, it was not worth looking at too many users before doing redevelopment. The trade-off was different for each project, but on average, just five users was optimal, with a slightly higher figure for large complex projects and smaller for simpler ones. The figure of “five users is enough” is often taken a little too literally without looking at the specific context of Nielsen and Landaur’s study, but the general lesson is that it is typically better to look at a relatively small number of users and fix the problems found thus far, before looking at more users.

As many systems are now deployed using web and agile methods in many projects, faster web-cycle times start to apply. The costs of performing modifications are usually quite low, with projects often having weekly or monthly update cycles. There is a temptation, especially with the web 2.0 philosophy of “perpetual beta,” to simply release products and let the users act as test subjects. While sometimes developers do ‘get away’ with this kind of practice, it is clearly not the way to build customer confidence! However, these fast cycle times do change the cost–benefit trade-off, and given it is easier and faster to fix problems immediately, this should lead to a culture of continuous improvement of user experience rather than a big-bang approach during initial development.
The Early Days: Visual Query Systems

Some of the earliest efforts to provide a user-centered approach to database management were the introduction of Visual Query Languages (VQLs). VQLs are languages for querying databases that use a visual representation to depict the domain of interest and express related requests. VQLs provide a language to express the queries in a visual format, and they are oriented towards a wide spectrum of users, especially novices who have limited computer expertise and who generally ignore the inner structure of the accessed database. Systems implementing VQLs are usually called Visual Query Systems (VQSs) (Catarci et al., 1997). In VQSs, the data representation language provides the user with a more natural view of the information, thus shortening the interpretation path that the user must perform to recognize the reality of interest from its computer-oriented representation. The same is true for the activity of expressing a query where direct manipulation of icons or navigation in diagrams substitute names of commands. Furthermore, visual languages are more flexible than traditional languages, where usually the learning of formal statements is needed to express even simple queries, each query being stated by a linear string where both operators and operands are expressed by words.

The advent of VQLs was due to several needs, including the following: providing a friendly human-computer interaction, allowing database search by non-technical users, and introducing a mechanism for comfortable navigation even in case of incomplete and ambiguous queries. It is worth noting that the real precursor of VQLs was Query-by-Example (QBE), proposed by Moshe Zloof in 1977 (Zloof, M., 1977). QBE was really ahead of its time. Indeed, Zloof’s paper states: “the formulation of a transaction should capture the user’s thought process….” This is a quite common idea today, but at that time (1977), the research on user interfaces and human-computer interaction was still in its infancy. It is worth noting that QBE is based not only on the usage of examples for expressing queries, but it also relies on the direct manipulation of relational tables inside a basic graphical user interface: an environment and an action modality that were quite unknown at that time. Another anticipatory idea is the incremental query formulation, i.e., “…the user can build up a query by augmenting it in a piecemeal fashion.” Many current papers still recommend allowing the user to express the query in several steps, by composing and refining the initial formulations (Catarci et al., 1997). Finally, many of the QBE ideas are still up-to-date, and it is amazing to note that QBE-like interfaces are nowadays adopted in commercial database systems, despite the current explosion of sophisticated visualizations and interaction mechanisms.
2. THE EARLY DAYS: VISUAL QUERY SYSTEMS

VQSs received the greatest attention by the database community only later on, during the '80s and early '90s. The main distinguishing characteristic of VQSs with respect to traditional query environments is that they are user-oriented. As such, they should be tailored to help users in performing the tasks they have in mind. Therefore, the characteristics of the classes of users who will be working with a particular interface and the tasks such users need to perform have to be well understood. Since the purpose of VQSs is to provide access to the information contained in a database, the main user tasks are understanding the database content, focusing on meaningful items, finding query patterns, and reasoning on the query result. These tasks require specific techniques to be effectively accomplished, and such techniques involve activities such as pointing, browsing, filtering, and zooming that are typically provided by a visual interface.

VQSs can be classified according to two main criteria. The first one is the visual representation that the VQS adopts to present the reality of interest, the applicable language operators, and the query result. The query representation is generally dependent on the database representation since the way in which the query operands (i.e., data in the database) are presented constrains the query representation.

The second criterion for the proposed VQS classification refers to the (visual) interaction strategies provided to retrieve data. Data retrieval through interaction with a VQS is usually accomplished through the following two main activities:

- Understanding the reality of interest. The goal of this activity is the precise identification of the fragment of the schema the query refers to. Generally, the schema is much richer than the subset of concepts that are involved in the query. The result of this step is a query subschema, i.e., the static representation of all schema items that are needed to solve the query.

- Formulating the query. The query subschema can be manipulated in several ways, according to which query operators are provided. The goal of query formulation is to formally express the operations and operands that eventually make up the query.

2.1 VISUAL REPRESENTATION

The concept of 'representation' captures the fact that a sign stands in for, and takes the place of, something else (Mitchell, W., 1995). Visual representation, in particular, refers to the special case when these signs are visual (as opposed to textual, mathematical, etc.) (Tufte, E., 1983, 1990). Note that visual signs often represent non-visual objects and abstract concepts. Moreover, the term 'representation' is often overloaded and is used to imply the actual process of connecting the two realms of the original items and of their representatives.

The main purpose of adopting a visual representation in a query system is to clearly communicate to the user the information content of the database(s), concentrating on essential features, and omitting unnecessary details. Such information is internally structured in several ways that mainly depend on the data model characteristics, but it must be rendered at the interface level in such a way that any user can easily grasp it. However, the interface visual representation has to be mapped in