Interactive Visualization of Large Data Sets

Parke Godfrey, Jarek Gryz and Piotr Lasek

Technical Report EECS-2015-03

March 31 2015

Department of Electrical Engineering and Computer Science
4700 Keele Street, Toronto, Ontario M3J 1P3 Canada

Interactive Visualization of Large Data Sets

Parke Godfrey, York University
Jarek Gryz, York University
Piotr Lasek, York University

TBA

Categories and Subject Descriptors: TBA [**TBA**]: TBA

General Terms: TBA

Additional Key Words and Phrases: TBA

ACM Reference Format:
Parke Godfrey, Jarek Gryz, Piotr Lasek, 2015. Interactive Visualization of Large Data Sets *ACM Trans. Embedd. Comput. Syst.* 9, 4, Article 39 (March 2010), 20 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

One picture is worth a thousand words. The idea of replacing a complex narrative with a single image may have become a cliche in journalism, but it is an absolute requirement in data exploration. After all, one may be able to read a thousand words, but cannot possibly look at, let alone understand, a billion data points. Understanding the data or, as some like to say, turning data into knowledge, may mean different things to different people (extracting structure, patterns, rules, constraints, etc.), but in all such cases visualization offers an indispensable tool in this effort. Indeed, visualization techniques can be applied at every step of data analysis, starting with initial exploration, through hypothesis generation, experimental validation up to the final presentation of discovery results. The path of exploration is by its very nature unpredictable, we may need to revise constantly *what* data is presented and *how* it is presented. Visual data exploration and analysis is interactive.

The distinction between interactive and non-interactive (call it passive) data visualization may seem a trivial one. After all, the process of interactive visualization is just a sequence of passive visualization steps with distinct data sets presented in different ways. In reality, however, the data sets accessed during this process are almost never distinct. The very point of interaction is to decide what data one wants to see in the next step based on what one has learned in the previous step. Most typically, one may want to see just a subset of the previous data (via selection or projection) or its aggregation. Some of the operations between the exploration steps became so common in the data analytics community that they acquired special names: for example, roll-up, drill-down, slice and dice, pivot. The fact that the data sets retrieved during

This work is supported by TBA
Author's addresses:

DOI: http://dx.doi.org/10.1145/0000000.0000000

the exploration process are related is of fundamental importance for the design of any interactive visualization tool.

The process of data visualization can be described from a high-level perspective as consisting of two simple steps: bringing the data into memory; then applying one of the visualization algorithm to this. There has been significant work on data visualization over the last 50 years. Interestingly, most of this work concentrated on the second step of the visualization process. This was understandable, as data sets were relatively small and the performance of the visualization tools was often determined by the efficiency of the graphics algorithms. Morever, if one wanted to show *all* data points on screen (even if it made little sense from the point of view of human perception), there were enough pixels to do so. With the advent of large data sets, whether in the form of data warehouses or scientific databases, a radical shift in the design of data visualization tools has to be made. First, we may no longer assume that raw data can be displayed on screen. The number of data points is now larger by orders of magnitude than the number of available pixels [Shneiderman 2008]. Data has to be compacted before any standard visualization techniques can be applied. We need *visual scalability*. Second, data retrieval and processing time now dominates the performance of the visualization process, so cannot be ignored. Without efficient database support, interactive visualization is impossible. Thus, we also need *data processing scalability*.

We limit the scope of this survey in specific ways. Indeed, writing a complete survey of computer-based visualization would be impossible to cover in a single paper. First, as made clear in the title, we are only interested in visualization of large data sets. To make it more concrete, let us fix ”large” to mean around a terabyte of data. This is reasonable, as many commercial data warehouses or scientific data sets are already beyond that size. We also do not discuss here the challenge of visualizing data sets of high dimensionality. This is another meaning of ”large” pointed out in [Heer and Kandel 2012]. Second, we focus on data stored in relational databases that is not domain specific (for example, geo-spatial or time-series)[1] Most of business data is natively in that form and many scientific databases, even if initially represented as graphs or XML, are transformed into and stored in relational format. By these two restrictions, our focus is on *database support* for visualization. We exclude in this survey work in visualization that does not explicitly address the issue of data processing scalability.

Last but not least, this paper is as much a reality check as it is a survey. Most researchers have assumed, and some of them still do, that ”visual analytics tools must support the fluent and flexible use of visualizations at rates resonant with the pace of human thought” [Heer and Shneiderman 2012]. In other words, for data exploration to be truly interactive, queries need to be responded to within a latency bound of 1–5 seconds [Kamat et al. 2014]. We must report that we have not encountered any system that would deliver this kind of performance under reasonable assumptions.[2] In fact, it has been observed that ”the appetite for data collection, storage, and analysis is outstripping Moore’s Law, meaning that the time required to analyze massive data sets is steadily growing” (Greg Papadopoulos, CTO of SUN, quoted in [Hellerstein et al. 1999]). Our message is not all bad news, however. Our reality check is that the community must re-calibrate its expectations in specific ways so that the expectations are achievable. We are at a point in data analytics and visualization research when we should reflect on what can and cannot be done in this area. We discuss these issues in Section 5).

[1]We list a few such systems in Section 5 without, however, discussing them in detail.

[2]By reasonable we mean an ad-hoc SQL query issued to a large database over a typical schema built within a sensible budget (say, under $10k).

The paper is organized as follows. In Section 2, we start with a overview of visualization techniques and systems developed over the last 30 years. Given the sheer amount of work done in the area, this is necessarily a subjective and high-level description of a subset of the relevant papers. Section 3 provides an overview of query optimization techniques developed mostly within a database community and then used in visualization systems. Then, in Section 4, we discuss data presentation challenge and conclude with some final thoughts in Section 5.

2. INTERACTIVE VISUALIZATION PREHISTORY

Over the last three hundred years inspiring data visualizations have been created. The famous Chart of Biography by Joseph Priestley from 1765 [Priestley 1765], Napoleon's Russian Campaign of 1812 [Minard 1812], and the map of Cholera Clusters in London in 1854 [Snow and Richardson 1965] which helped to identify the sources of water contamination are three such. These and other historically important visualizations [Tufte 1990] proved to be of great importance in the field of data analysis. They were successful because a user was able to intuitively understand the graphical representation of data to easily draw valuable conclusions.

Nevertheless much work has been done in order to create visualization in a comprehensive way. If the visualizations designer wants to pass certain knowledge about data to the perceiver, the semiotics approach by Bertin needs to be considered. In his Semiology of Graphics [Bertin 1983], he addresses the different issues related to the process of creation of a good visualization. He says that a designer should understand a system of related information, be able to create a mapping from data to its visual representation, present the visual representation on a computer screen, and provide appropriate methods of *interacting* with the visual representation that could include methods for varying the presentation. He should be also able to verify usefulness of the representation and its interaction methods. Bertin bridged the world of data with the world of graphics by connecting a number of basic graphical shapes with the types of knowledge they could represent. For example, he believed that points were best for representing location, lines were best to express a measurable length, boundaries, routes or connections whereas areas signified something important on the plane that had a measurable size. Bertin specified and described in details numerous types of visual variables such as position, size, shape, value, orientation, color, texture and motion. This set could be easily extended and adapted by using other visual variables such as motion, saturation, flicker, depth, illumination and transparency. All of these variables may have their own features and attributes (e.g., saturation intensity) which could be altered to better reflect the data the variable represents. Additionally, the variables can be combined into more complex constructs, for example charts, diagrams, networks, maps or symbols. Bertin's work was the first attempt to provide theoretical foundations to contemporary data visualizations. A great majority of existing tools still employs concepts described almost sixty years ago in the first edition of his Semiology of Graphics.

The advent of database management systems brought automation to storing and accessing digital data. This created possibilities to visualize large amounts of data efficiently. For the first systems designed for data visualizations Bertin's work and the idea of mapping data into visual variables was useful. For example, in CHART [Benson and Kitous 1977] which was a simple data analysis and report design program, a mechanism for mapping numerical data to graphic variables was used. Rows and columns could be re-organized by means of different operators such as ranking, sequencing and grouping, as well as re-computed from arithmetic combinations of existing rows and columns. Another system which used concepts presented by Bertin such as mapping a data object to visual variables and which was designed to work with data stored in a relational database was developed in 1986 [Mackinlay 1986]. The goal of the tool called

APT (A Presentation Tool) was to develop an application independent system which would be capable of creating automatically visual representations of relational data. In order to achieve good results, the authors of APT codified graphic design elements and made the assumption that graphical presentations (visualizations) are sentences in a graphical language. They defined additionally a concept of *expressiveness* and *effectiveness*, which were likely inspired by the Bertin's idea of usefulness of graphical representation of data. Expressiveness can be intuitively understood as an ability to express a set of facts by means of a given language; effectiveness is related to the ability of a viewer to understand a given graphical representation. Formalization of graphical sentences (visualization) allowed APT to determine to what extent a given visualization meets the expressiveness and effectiveness criteria. A similar approach (in terms of automated determination of effectiveness of produced graphics as well as for defining visualization goals by means of a logical language) was used in BOZ [Casner 1991]. This tool employed a task-analytic approach which meant that users could specify a logical description of a visualization task to be performed. The system analyzed the task and chose an optimized way to generate the results. The system also supported *interactive* manipulations of the graphical objects representing the data.

The early tools were able to support automatically a process of visualization generation but many ideas related to creating better and more understandable visualizations remained unimplemented. Mackinley, for example, considered animation and 3-D presentation as means which could used in the process of data visualization. What is even more interesting perhaps, is that researchers noticed a need for designing their systems so that they were *interactive*. Nevertheless, the interactivity in the early eighties and nineties was only considered as an ability to generate visualizations automatically, or semi-automatically, based on a special visualization query language or a graphical representation of a traditional SQL query. Shneiderman's mantra *Overview first, zoom and filter, and then details on demand...* [Shneiderman 1996] in most of the systems was implemented so that each of its steps was actually generated by a separate query issued to the database system.

Subsequent research efforts focused on generating graphics using application-independent design knowledge. For example, in the case of the SAGE system [Roth and Mattis 1991], the design knowledge module was composed of two components: a library of presentation techniques (techniques such as tables, charts, maps, network diagrams; information connecting types of data with suitable technique; syntactic or structural relations among elements such as axes, lines, points or labels), and mechanisms for selecting and combining those techniques. With SAGE, it was possible to automatically generate presentations of information and design displays with complex combinations of data by *interactively* changing the so-called presentation goals, which could be specified by a system's operator in a form of a special language. A user could specify relations (such as Has-Part and Responsible-For), objects (simply tables from a database) and presentation goals (such as Show-Correlation and Locate-Easily). Other systems based on SAGE used similar approaches of semi-interactive exploration of databases. IDES [Goldstein et al. 1994], for example, aimed directly at similar knowledge-based interactive data exploration and tried to overcome the limitations of existing systems with complex and difficult to learn query mechanisms that still did not cover all the operations required by users. It integrated work on SAGE, and extended this with a prototypical graphical interactive manipulation component. Nevertheless, the concept of interactive data exploration was rather naive, by means of workspaces with different widgets such as buttons, sliders, combo boxes, tables, and an aggregate manipulator by which a user was able to generate, execute, and reissue queries. At the end, the user received a corresponding data visualization view. If the result was not satisfactory, the user could adjust settings of the widgets to re-

peat the whole process. The dynamic queries allowed drag-and-drop construction of queries, which allowed users to focus more on the process of data exploration rather than on the tools. IDES was capable of changing *granularity* of the data by *aggregation* (creating meaningful groups of data objects) or by *decomposition* (dividing larger data groups into smaller ones). Similarily, Keim and Kriegen in VisDB [Keim and Kriegel 1994] noticed the possibility of arranging data objects or dimensions into groups, even though they designed their system so that each display pixel represented one database item. Experiments which they performed on geographical data led them to formulate another problems: for example, how to deal with data that do not have natural representation as a map; how to fit large data into small screen; and how to find the best highlighting methods such as points, colours, flicker, and light.

Another step forward to more interactive visualization and exploration was the idea of using stored results of visualization in a form of a *slide show* (Visage [Roth et al. 1996]). Slides were created by a user by dragging and dropping desired graphics onto a special frame. A user had an option to come back to those stored visualizations at any time. Further research led to creation of Visual Query Language (VQE) [Derthick et al. 1997] which added capabilities of direct manipulation and exploration of databases to Visage. By means of this language it was possible to dynamically link queries and visualizations so that operations on visualizations updated the data and vice versa - if data was changed, the visualization changed automatically. Some systems were even designed so that they supported building queries by means of a specialized graphical language. Such a language was used in InfoCrystal [Spoerri 1993]. In this case, however, a graphical language was used both for defining queries and visualizing results. Its structure composed of so-called crystals based on Venn's diagrams and the elements of graphical queries could be combined into complex blocks and organized hierarchically. In case of larger data sets (with the number of tuples much greater than number of a screens pixels) the systems (especially Visage) had functions for *dynamic data aggregation*. With Visage it was possible to *aggregate* a set of data tuples into a new tuple having properties *derived* from its elements. The family of SAGE systems and solutions (SAGE, IDES, Visage, VQE) was commercialized and evolved into CoMotion [Chuah and Roth 2003] (a product enabling data sharing, visualization and messaging) and later into Command Post of the Future (CPOF) [Chuah and Roth 2003] a software allowing military commanders to manage a battlefield.

Some other systems put more attention to design more flexible graphical user interface so that users could perform a number of visual operations such as zooming, 3-D manipulation, panning, filtering and selection of details. Those systems were also *interactive* thanks to numerous sliders (similarly to [Ahlberg and Wistrand 1995]). For example VIS [Ahlberg and Shneiderman 1994b] and IVEE [Ahlberg and Wistrand 1995] and eventually Spotfire [Ahlberg 1996] were designed so that they used the concepts of dynamic query filters (allowing users adjust query parameters by means of sliders), starfield displays (scatter plots with additional features such as selection or zooming) and tight coupling (an idea of using a query result as an input to produce another query to support progressive query refinement [Ahlberg and Shneiderman 1994a]). Later, a concept of dynamic queries [Ahlberg et al. 1992] was introduced. Dynamic queries allowed users to formulate queries using graphical widgets called query devices (e.g., rangesliders, alphasliders, and toggles). Spotfire worked so that it attached a graphical object to each object from the database. However, in order to achieve appropriate performance in some cases it had to use approximations when rendering visualization. For example, it might have to render objects at a lower resolution, display complex objects as wire frame models, skip textual labels, not fully redraw the screen while performing time-expensive computations. Nevertheless, the necessity of using approximations was prompted not by the size of a dataset but rather by limi-

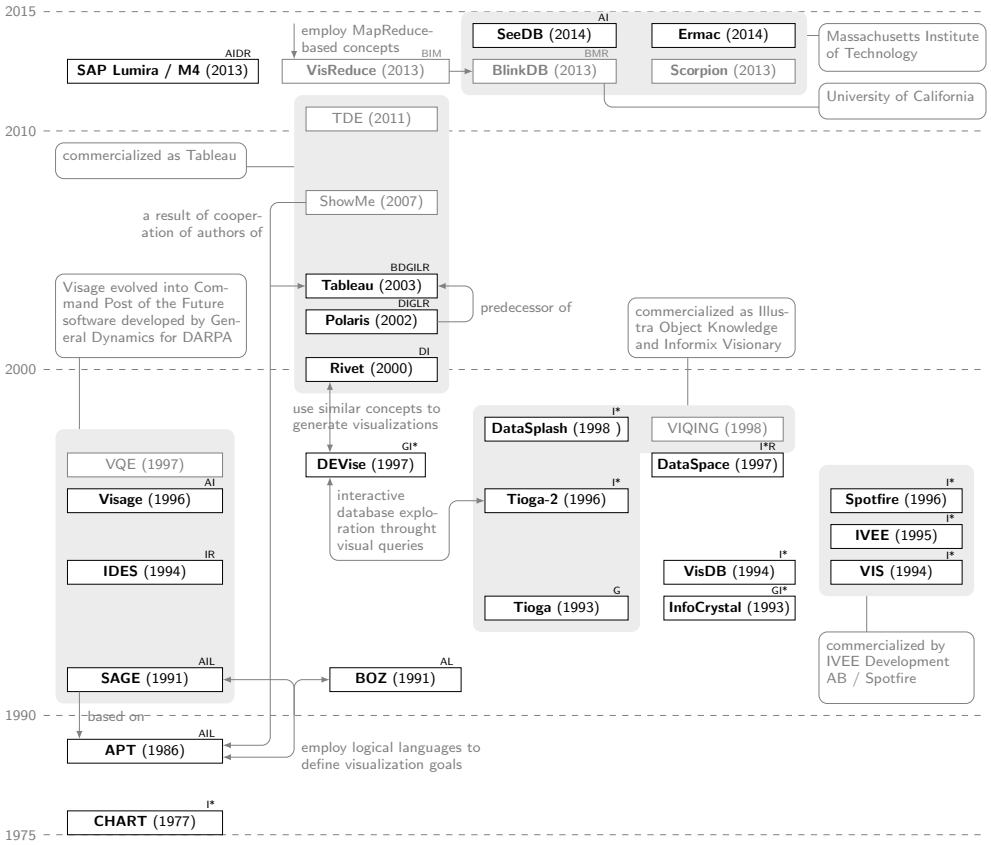

Fig. 1. A map of database visualization tools covering the years 1986 to 2014. (A - generates visualizations automatically, B - operates on Big Data, D - uses an integrated database or a table-like structures to store imported data, G - uses a graphical language to define visualizations, I - supports interaction (* - naively, e.g. by), L - uses a special language to define visualizations, M - uses a concept of Map-Reduce to deal with big data, R - has a function to aggregate data; gray area denotes a family of systems developed by the same team; elements within a family of systems marked with gray color do not fulfill our database visualization tool criteria, however were important with respect to the other tools in the group.

tations of graphics hardware. Similarly, the progressive refinement was implemented using widgets (sliders, buttons, etc.) and worked so that each change of a widget's state triggered another query which result could be used to refine it by properly adjusting the widgets. So the process of query refinement was not optimized, did not use cashed results and in order to get results the data had to be retrieved again. As regards 3-D manipulation mentioned above, the DataSpace [Petajan et al. 1997] system needs some attention. It was a mouse-based 3-D navigation tool which allowed user to zoom-in and zoom-out graphically presented data. Additionally it incorporated a variety of techniques such as: aggregation, data drill down, multidimensional scaling, variable transparency and query by example.

Another approach used in VisDB [Keim and Kriegel 1994] but not common in other visualization tools took advantage of the fact that in many cases only a limited number of attributes are of interest so the number of visualised dimensions was restricted to those which were part of the query. This tool was also capable of visualizing not only tuples fulfilling query criteria, but also the approximate results by determining tuples

similar (in terms of a distance measure) to those returned as a result. Data in VisDB were visualized so that the system first sorted them with respect to their relevance to a specified query. Next, the relevance factors were mapped to appropriate colors. However, similarly to the other tools, in order to support interactivity, the system basically *recalculated* visualization after each modification of the query by means of a graphical interface.

The idea of incremental generation of visualizations was common in early database visualization systems. While some systems were using sliders, buttons, text boxes and other widgets, Tioga [Stonebraker et al. 1993] and Tioga-2 [Aiken et al. 1996] introduced *recipes* as definitions of a whole process of data visualization. A recipe was constructed by a user in a form of a graph in which each node represented a single step. A user could execute such a recipe *interactively* changing parameters of different steps based on obtained visualization. The authors of Tioga-2 also noticed that in case the data is *aggregated* or some areas of a dataset need more attention it could be useful to use a mechanism of zooming in or drilling down in order to see more details. To that effect the mechanism of *wormholes* was introduced and worked so that a user could move from one canvas to another as zooming in. If a user wanted to go back to the previous view, in was possible with a rear view mirror which was used to illuminate wormholes back to the starting canvas from which user started zooming in. Wormholes and rear view mirrors were later replaced by *portals* (subareas of a canvas used to display other canvases) in DataSplash [Woodruff et al. 2001]. This family of systems used a special graphical environment for defining queries called VIQING [Olsten et al. 1998], which provided a visual interactive interface for query specification.

The general approach in the 1990s was to represent a single database object by a single instance of a graphical variable. However, the DEVise system [Livny et al. 1997a] introduced the idea of construction a visualization view employing three different layers such as: a background (on which a visualization was drawn), a data display (for graphical objects representing data objects) and an additional cursor display (a data-independent layer used for example for highlighting a portion of the data).

Tools designed in early 2000s put even more attention on *interactive* visual data analysis. For example, in case of Rivet [Bosch et al. 2000] the internal database structure was designed to support the rapid development of interactive visualization of large data. Data were imported to the system and stored in Rivet in a form of *tuples*, which Rivet considered as unordered collections of attributes which could be grouped into tables if they were of the same format. Rivet was capable of supporting different types of data sources such as data bases and files. Rivet used a homogeneous data model, separation of data objects from visual objects, a mechanism of selectors and visual *metaphors* (functions translating data objects into their graphical representations) to visualize selected portions of data. Expertise gained during development of Rivet led to formulation of VisQL - a language for the Polaris system [Stolte et al. 2002] which used a concept of *shelves* corresponding to tables' attributes. It facilitated the generation of precise sets of relational queries directly from visual specifications represented by a given configuration of fields. It also supported interactive visualization by using techniques such as: deriving additional fields, sorting and filtering, brushing and tool-tips, and undo and redo functions. Additional fields introduced in Polaris allowed users to see additional information derived from a data set such as *aggregated* quantitative measures, a count of distinct values, ad hoc grouping, and threshold aggregates. Soon after Polaris was created, Tableau Software was founded and Polaris became the first product of the company.

3. DATA PROCESSING CHALLENGE

Almost all of the systems described in the previous section were designed to work with just megabytes of data. They did not need any special data processing or data presentation techniques to allow truly interactive visualization. Over the last decade or so, however, our ability to collect and store data has grown faster than our ability to process it. Scalability has become the key challenge for visual analytics. Indeed, keeping up with ever increasingly large data sets has been an uphill battle in many other areas of computer science. The database community has been hard at work to find solutions to the challenge for many years now. Some of them were hardware based (e.g., parallelization, increased bandwidth, and clever storage schemes), but most involved new algorithms or even new paradigms for query processing.

In this section we will review major trends in this area with examples–wherever possible–of implemented visualization systems. To organize these trends, we divided research in this area into four major domains along two orthogonal dimensions as shown in Table 2. The first dimension refers to the *type of data* against which queries are executed, that is, whether the data is pre-processed or not. The second dimension refers to the *type of answers* expected, that is, whether they are supposed to be exact or just approximate. Within each of the dimensions some other subdivisions can be identified; they will be discussed in more detail below.

3.1. Exact answers

3.1.1. Exact answers from raw data. The paradigm of query processing in database systems has always been the *batch* approach (represented as the top-left quadrant in in Table 2). The user issues queries, the system processes them silently for some time, and then the system returns an exact answer. Many tools have been designed over the last 30 years to speed up query processing time, but the default has been to process queries against the raw data, that is, individual records stored in relational tables. Not surprisingly, this has been the least efficient way, in terms of latency, of providing input to a visualization tool. Indeed, most of research in database query processing over the last years has been to *move away* from that paradigm either by providing a user with approximate answers only or by preprocessing the data to improve query performance. (These are the three remaining quadrants in Table 2.)

But the batch approach has one important advantage over *all* other systems described below: it allows the user to issue a truly ad hoc query and get an exact answer to it. In the era of the ”big data” this may be considered a luxury, but there are scenarios where this approach is necessary. The Dremel system, which has been in production at Google since 2006, provides all advantages of the batch approach with the latency expected of an interactive system. Dremel puts together a number of well-known techniques: parallelism on a shared cluster of commodity machines, columnal storage, and data compression to process terabytes of data in a matter of seconds. It is unlikely, however, that a system like Dremel will find its way into a wider market: the cost of the system (3000 machines in a cluster), and a proprietary query language and query processing scheme put it out of reach but for the largest enterprises.

3.1.2. Exact answers from preprocessed data. The idea of data preprocessing is simple: rather than executing a query at runtime, do it in advance and save the results for future use. The concept of *materialized view* (which is in fact a misnomer as it is really a materialized query result) had been introduced within a database community as early as 1980s (see [Ashish Gupta 1999] for an overview). Like a cache, a materialized view provided fast access to data; the performance difference may be critical in applications where queries are complex or need to retrieve large amount of data. In a data warehouse, where pretty much all queries involve aggregation, such pre-aggregated

		Exact answers	Approximate answers	
			Static	Incremental
Raw data		Dremel [Melnik et al. 2010]		Control [Hellerstein et al. 1999], sampleAction ([Fisher et al. 2012a], [Fisher et al. 2012b]), VisReduce [Im et al. 2013], [Jermaine et al. 2006]
Preprocessed data	Static	inMens [Liu et al. 2013]	[Li et al. 2008], [Chaudhuri et al. 2007], AQUA [Acharya et al. 1999a] and [Acharya et al. 1999b]	BlinkDb [Agarwal et al. 2013]
	Dynamic	XmdvTool [Doshi et al. 2003], ATLAS [Chan et al. 2008]	[Chaudhuri et al. 1999]	DICE [Kamat et al. 2014]

Fig. 2. Categorization of data processing paradigms with some representative systems. Visualization systems are in red.

materialized views have been called 'cubes' [Gray et al. 1996]. Again, the idea was that rather than process the queries as they arrive, typical aggregations (for example, total sale value grouped by product, location and time) should be preprocessed in advance and stored in a warehouse along with the raw data. Such aggregations can be linked directly to their visual display for more efficient interaction at runtime. This idea has been explored in [Liu et al. 2013].

The advantages of data preprocessing are as obvious as the limitations: queries can be answered fast as long as *their* answers have been previously stored[3]. If a cube contains total sales per state, it cannot be used to answer a query that asks for sales per city. For applications where a set of possible queries is fixed this is not an issue, but for interactive exploration of data, static pre-processing is of limited use.

Not surprisingly, few systems (the Hotmap project [Fisher 2007] being one of the exceptions) used static data pre-processing. But queries do not have to be preprocessed off-line; that is, before the user starts her query session. Instead, once the user starts asking queries, future queries can be predicted and processed in the background. The idea of dynamic data preprocessing is based on a few clever observations [Doshi et al. 2003]. First, visualization tools limit - to some extent - the types of queries that are asked. Queries tend to be contiguous rather than entirely ad hoc as the visual interface provides controlled means of expressing navigational requests. Second, a user tends to look around a particular region (defined geographically, chronologically, or along some other dimension) before moving to another region. In other words, the user navigation tends to be composed of several small and local movements rather than major and unrelated movements. Third, since the user will be examining the visual displays, there typically would be delays between user queries. The first two observations suggest a certain level of predictability of future queries; the last one offers time to precompute

[3]This is not quite correct: a query may be answered from a combination of materialized views even when it cannot be answered from any of them individually. But even in this case, there must be views that are related in very specific ways to the query

these queries. The XmdvTool [Doshi et al. 2003] offers an array of speculative prefetching strategies. When the system is idle, a prefetcher will bring data into the cache that is likely to be used next. In addition to prefetching XmdvTool is also using semantic caching techniques which group data in cache with respect to their semantic locality (for example, proximity in time or distance) rather than recency of their use.

Of course, the performance of a system based on query pre-fetching depends on the level of predictability of future queries. In some domains user interactions will be naturally limited. For example, the *Atlas* system [Chan et al. 2008] designed to store large historical time-series data, allows only six directions of exploration (pan left, pan right, scroll up, scroll down, zoom in, zoom out). The predictive algorithm is based on observing that there is a sense of momentum associated with the direction of exploration. For example, if an analyst is panning to the left at time t, she is likely to continue panning left at time $t+1$. It is worth noting that pre-fetching not only speeds up query processing, but also makes the process of visual analysis less disruptive from the HCI point of view.

3.2. Approximate Answers

As discussed in Section 1, the amount of data to be visualized often exceeds by orders of magnitude the number of pixels available on a display. The data has to be reduced (by filtering, aggregation, principal component analysis, etc.) as only a small portion of it can be displayed. Since each of the reduction techniques causes a loss of detail, the "visual answers" can only be approximate. But if we can no longer show exact answers to our queries, perhaps the answers retrieved from a database should be approximate as well. Indeed, most of the research in visualization systems over the last few years focused on computing approximate rather than exact answers. There are two primary ways to achieve that: statically, when queries are computed over preprocessed samples of data or dynamically through incremental (online) query processing.

3.2.1. Sampling. Sampling has been used routinely in database systems. IBM's DB2, for example, supports the `tablesample` operator that can be applied to base tables to return a desired fraction of tuples. Thus, it may seem that instead of running queries on full tables one may access only their samples to achieve an appropriate balance of processing time and answer precision. Unfortunately, most of SQL operators do not commute with sampling, that is, a uniform random sample of a query result cannot be produced from a uniform random samples (no matter how large) of the tables used in a query. Consider the following example [Chaudhuri et al. 1999].

Example 3.1. Let the query be $Q = R \bowtie S$, where: $R(A, B) = \{(a_1, b_0), (a_2, b_1), (a_2, b_2), (a_2, b_3), \ldots, (a_2, b_k)\}$ and $S(A, C) = \{(a_2, c_0), (a_1, c_1), (a_1, c_2), (a_1, c_3), \ldots, (a_1, c_l)\}$. Given any samples of R and S, it is impossible to generate a random sample of Q for any reasonable sampling fraction or under any reasonable sampling semantics.

Similar examples of non-commutativity can be provided for `select distinct`, `group by`, `min`, `max`, and other typical SQL operators. Nested queries pose yet another challenge. If a nested query returns a value used in a selection condition of the outer query, that value has to be computed precisely. Otherwise the query result is meaningless.

To overcome this problem two solutions have been proposed. One is to pre-process the data in a certain way to make the sampling useful for query processing. For the example above, this might require including a_1 and a_2 in samples of R and S respectively. The second solution is to design algorithms - mostly for joins - that would be immune to the problem discussed above. We should emphasize, however, that it is impossible to provide a meaningful approximate answer to *every* SQL query with only a fraction of the data processed. In other words, there is a limit to what approximate query an-

swering can deliver, a limit that is often ignored or overlooked. Many papers reviewed here do not even specify what types of SQL queries their systems can handle.

3.2.2. Approximate Incremental Answers. As we pointed out in Section 1, data analysis is fundamentally an interactive process in which a user issues a query, receives a response, formulates the next query based on the response, and repeats. People naturally start by asking broad, big-picture questions and then continually refine their questions based on feedback. They do not need exact answers to such questions, but the do expect rapid results. They also want control over the precision of the answers. Interactive systems should produce continuously refining approximate answers and when such answers are good enough the user should be able to stop the process and move on.

The most straightforward approach to incremental visualization has been implemented in *VisReduce* [Im et al. 2013]. The system is in fact similar to *Dremel* described above: it uses columnar storage combined with slightly modified MapReduce approach. But rather than computing complete answers (as in *Dremel*) *VisReduce* incrementally returns partial answers computed over progressively larger samples of the data. A similar idea is explored in the *sampleAction* system [Fisher et al. 2012a; Fisher et al. 2012b], but here a user is also provided with a confidence bound (error measure) for the returned results. Unfortunately, the descriptions of both systems are not detailed enough to determine how they circumvent the problems associated with sampling.

A more sophisticated approach has been used in the *Control* project [Hellerstein et al. 1999]. Here, the first specialized techniques for joins over samples, called ripple join algorithms, were introduced [Haas and Hellerstein 1999]. The idea was to adjust the sampling rates over each of the tables dynamically during the join based on the data seen so far. Similar algorithms have been also proposed in [Jermaine et al. 2006]. Still, even these more sophisticated join algorithms cannot handle extreme data skew described in Example 3.1.

The DICE system [Kamat et al. 2014] combines sampling with speculative query execution (similar to the XmdvTool) to achieve interactive latencies for data cube exploration.

3.2.3. Static Approximate Answers. To address this and other problems of non-commutativity of sampling with SQL operators, the idea of precomputed samples, or *sample synopses*, was introduced. Rather than sampling base tables at runtime, we can pre-compute certain carefully crafted samples and store them for future use [Acharya et al. 1999c; Acharya et al. 2000; Gryz et al. 2004; Chaudhuri et al. 2007]. These samples are designed to be used only with specific queries so that the queries executed over the samples are guaranteed to return answers of an arbitrary precision. Also, the cost of storing the samples - compared to the traditional materialized views - is negligible. However, just as in the case of materialized views, we sacrifice flexibility: not every query can be answered using the stored samples. And this is not a question of precision as there will always be queries that cannot be answer with *any* precision (unless samples of all possible queries are stored).

Example 3.2. Consider the following SQL query:

```
select a, max(b)
from  R
where c=X
group by a
```

The groups returned by the query and the maximum value of attribute b in each of the groups depends on the value of c. Unless we store samples of this query for *all* values of c, no meaningful answer based on sampling is possible.

We should also note that since sample synopses are pre-processed statically off-line they are of limited value for incremental visualization as they can only provide one approximate answer of a fixed precision (storing multiple samples of the same data is not feasible in practice). One way to avoid this problem is to pick an appropriately sized *sub-sample* of a stored sample based on the query's required response time or precision constraints [Agarwal et al. 2013].

The idea of materializing samples can be pushed even further to build the entire database out of them. This idea has been explored in [Li et al. 2008].

3.3. Tightly-Coupled Systems

Most visualizations systems described above retrieve data from a database first and then use specialized visualization tools to render it. This decoupled approach results in significant duplication of functionality and misses tremendous opportunities for cross-optimization. The idea of *integrating* a database system with a visualization systems seems self-evident, yet the exact level and juncture of integration has been understood differently by different people.[4]

Probably, the first attempt to build a tightly-coupled database/visualization system was the DEVise system [Livny et al. 1997b]. The emphasis there was on integrating querying with data visualization features: users could render their data in a flexible easy-to-use manner. Mapping visual operations to data access makes query optimization more effective as the semantics of how different parts of visual presentations are linked offered hints on what to index, materialize or cache.

The idea of mapping visual operations to database queries has been explored and implemented in various ways in many systems since then (the most notable implementations are [Doshi et al. 2003] and [Chan et al. 2008] discussed above). But communicating to the database what the user wants to see may help in other ways than just query performance. A visualization tool may also tell the database *how much* data it needs to render a picture thus limiting the amount of data sent from a database. This improves performance at two levels: it reduces the communication costs and eliminates the need for data reduction at a later stage. The M4 system [Jugel et al. 2014] implemented in SAP Lumira addresses exactly this problem. Rather than executing a query as given, M4 relies on the parameters of the desired visualization to rewrite the query. Then, it develops an appropriate visualization-driven aggregation that only selects data points that are necessary to draw the complete visualization of the complete underlying data. A similar ideas have been implemented in the ScalaR system [Battle et al. 2013].

Recently, a call for even tighter integration of a database and visualization systems has been made [Wu et al. 2014]. The decoupled approach has three major drawbacks. First, the database is unaware of related queries. Second, visualization tools duplicate basic database operations. Third, visualization tools assume that all data fits entirely in memory. To alleviate these problems the authors advocate building a Data Visualization Management System, a system that would make all database features available for visualizations (being a vision paper, no specific solutions - other than possible research directions - are provided).

[4]The distinction between coupled and decoupled systems adds yet another dimension to the categorization in Table 2. We did not include it there not only because it is hard to visualize, but also because it is not a binary property of any system.

It is interesting to see how the data processing challenge was addressed in a commercial data visualization system. Tableau Software is likely the most successful among them. The company delivers a number of data visualization tools suited for business intelligence such as: Tableau Desktop, Tableau Server, Tableau Online, Tableau Reader and Tableau Public. The early architecture of Tableau Desktop was designed so that it was capable to connect to different relational as well as hierarchical databases. In order to reduce the load (in terms of processing large data sets) it uses data *extracts* (a filtered or sampled subsets of original data set) which were originally processed using the Firebird open source database. However, even with extracts generated based on filtered, sampled or rolled up subsets of the original data set, Firebird turned out not to be efficient enough and Tableau Software decided to create its own read-only column-based data engine (TDE) optimized for data visualization [Wesley et al. 2011]. Still, further growth of sizes required further improvements of TDE's efficiency by leveraging compression techniques [Wesley and Terlecki 2014] so that Tableau products could be more interactive. As a result Tableau's in-memory database (also implemented in Vertica and PowerPivot) scales up to interactive queries across millions of rows. Beyond this range, however, we are back to the fundamental issue: a database simply cannot produce a full response to a query in interactive time [Fisher 2011].

4. DATA PRESENTATION CHALLENGE

Visual scalability is the capability of a system to effectively display large data sets in terms of either the dimension of data points (usually understood as the number of attributes to be presented) or the sheer number of these points. The issue of presenting multiple dimensions on a 2-d display has been with us for a while now; the second problem is relatively new. In most realistic visualization systems the amount of data to be visualized exceeds the number of pixels of display by orders of magnitude. The data has to be reduced or compressed in some way before it can be displayed. As the rate of compression is increasing, more and more details of the actual data will be lost. Thus, the data reduction process must be followed by an appropriate presentation of the modified data.

The data reduction process can be performed by the database system or by specialized algorithms tied to the rendering tools. There are three main methods for reducing data within a database: filtering, aggregation, and sampling [Battle et al. 2013].

Filtering is the most straightforward method: rather than presenting the complete data set, only a subset of the data points is selected (using the `where` clause in SQL) for display. Filtering does not require any specialized visualization methods as the original data points are presented. An obvious disadvantage of filtering is its inability of showing the complete data set.

Aggregation groups data into subsets (usually performed via the `group by` clause in SQL) and returns summaries of the groups as *sum*, *average*, etc. At the presentation level, aggregations require new visualization techniques as individual points are no longer displayed. [Elmqvist and Fekete 2010] provides a comprehensive overview of rendering techniques for displaying aggregated data.

Sampling (which is supported by most database systems) returns a fraction of the original data points given some specified probability. In this sense, the answer produced by sampling is approximate and represents uncertain information. In general, uncertain information can be specified in three different ways [Streit et al. 2008]: estimates (the values are known to be inaccurate with unknown precision), intervals (the value is known to fall within a specific range), and probabilities (the value can be expressed as a probability curve). It is a challenge to display uncertain data in a way easily readable to users; there is no straightforward solution to it from the HCI perspective. Of course, the problem is not specific to large data sets; the reader is referred

to [Olston and Mackinlay 2002; Kosara et al. 2001; Wittenbrink et al. 1996; Sanyal et al. 2009] for more discussion.

5. CONCLUSIONS

Visualization provides a powerful means of making sense of data. Visual analysis typically progresses in an iterative process of view creation, exploration, and refinement. To be most effective, visual analytics tools must support the fluent and flexible use of visualizations at rates resonant with the pace of human though [Heer and Shneiderman 2012]. But this appetite for visual data analysis will most likely remain insatiable. Computing power has not kept pace with the growth of digital data and there is no sign that this will change any time soon. It is unlikely, that a *general purpose* visualization system can provide smooth interaction over large data sets (we have not seen such a system so far).

What then can we get instead? What constraints do we have to impose on an architecture of a system to provide truly interactive visualization? The answers to these questions are in fact provided in the papers reviewed here. In all cases when the reported latency was within the limits expected of an interactive performance, one or more of the following constraints were imposed upon a respective system:

— The data set is small (often in single gigabytes). Although this condition disregards the call for visualization of *large* data sets, it is acceptable for most of the typical application in real world.
— A system is built for a very specific type of data, for example, time-series only. Limiting the type of data to be visualized often simplifies the types of queries (even if it is not stated explicitly) that a user can ask, thus making their execution more efficient.
— Queries are processed over samples of data rather than full database. This has been a path chosen in most of the recent systems as it provides truly interactive performance. There are two problems with this approach. The first, already discussed in Section 3, is the limit on the types of queries that can be meaningfully asked against a database. The second, is the inability of discovering outliers in data (unless special provisions are made in sampling techniques) which for some applications may be indispensable.
— Data is reprocessed (for example, by storing materialized views). This is a method routinely used in OLAP. Unfortunately, it does not allow for ad-hoc queries.
— Massively parallel systems. This is the only approach that works. But it is costly (3000 machines in a cluster were used to build Dremel).

REFERENCES

Swarup Acharya, Phillip B. Gibbons, and Viswanath Poosala. 2000. Congressional Samples for Approximate Answering of Group-By Queries. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein (Eds.). ACM, 487–498. DOI:http://dx.doi.org/10.1145/342009.335450

Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. 1999a. Join Synopses for Approximate Query Answering. In *Proceedings of SIGMOD*. 275–286.

Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. 1999b. Join Synopses for Approximate Query Answering. In *Proceedings SIGMOD*. 275–286.

Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. 1999c. Join Synopses for Approximate Query Answering, See Delis et al. [1999], 275–286. DOI:http://dx.doi.org/10.1145/304182.304207

Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. BlinkDB: queries with bounded errors and bounded response times on very large data. In *Eighth Eurosys Conference 2013, EuroSys '13, Prague, Czech Republic, April 14-17, 2013*, Zdenek Hanzálek, Hermann Härtig, Miguel Castro, and M. Frans Kaashoek (Eds.). ACM, 29–42. DOI:http://dx.doi.org/10.1145/2465351.2465355

Christopher Ahlberg. 1996. Spotfire: an information exploration environment. *ACM SIGMOD Record* 25, 4 (1996), 25–29.

Christopher Ahlberg and Ben Shneiderman. 1994a. The alphaslider: a compact and rapid selector. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 365–371.

Christopher Ahlberg and Ben Shneiderman. 1994b. Visual information seeking: tight coupling of dynamic query filters with starfield displays. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 313–317.

Christopher Ahlberg, Christopher Williamson, and Ben Shneiderman. 1992. Dynamic queries for information exploration: An implementation and evaluation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 619–626.

Christopher Ahlberg and Erik Wistrand. 1995. IVEE: An information visualization and exploration environment. In *Information Visualization, 1995. Proceedings.* IEEE, 66–73.

Alexander Aiken, Jolly Chen, Michael Stonebraker, and Allison Woodruff. 1996. Tioga-2: A direct manipulation database visualization environment. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 208–208.

Inderpal Singh Mumick Ashish Gupta (Ed.). 1999. *Materialized Views*. MIT Press.

Duane A Bailey, Janice E Cuny, and Craig P Loomis. 1990. Paragraph: Graph editor support for parallel programming environments. *International Journal of Parallel Programming* 19, 2 (1990), 75–110.

Leilani Battle, Michael Stonebraker, and Remco Chang. 2013. Dynamic reduction of query result sets for interactive visualizaton, See Hu et al. [2013], 1–8. DOI:http://dx.doi.org/10.1109/BigData.2013.6691708

Thomas Baudel. 2004. Browsing through an information visualization design space. In *CHI'04 Extended Abstracts on Human Factors in Computing Systems*. ACM, 765–766.

Steve Benford, Dave Snowdon, Chris Greenhalgh, Rob Ingram, Ian Knox, and Chris Brown. 1995. VR-VIBE: A Virtual Environment for Co-operative Information Retrieval. In *Computer Graphics Forum*, Vol. 14. Wiley Online Library, 349–360.

William H Benson and Bernard Kitous. 1977. Interactive analysis and display of tabular data. *ACM SIGGRAPH Computer Graphics* 11, 2 (1977), 48–53.

Jacques Bertin. 1983. *Semiology of Graphics*. University of Wisconsin Press.

Wes Bethel, Cristina Siegerist, John Shalf, Praveenkumar Shetty, TJ Jankun-Kelly, Oliver Kreylos, and Kwan-Liu Ma. 2003. VisPortal: Deploying grid-enabled visualization tools through a web-portal interface. *Lawrence Berkeley National Laboratory* (2003).

Robert Bosch, Chris Stolte, Diane Tang, John Gerth, Mendel Rosenblum, and Pat Hanrahan. 2000. Rivet: A flexible environment for computer systems visualization. *ACM SIGGRAPH Computer Graphics* 34, 1 (2000), 68–73.

Michael Bostock and Jeffrey Heer. 2009. Protovis: A graphical toolkit for visualization. *Visualization and Computer Graphics, IEEE Transactions on* 15, 6 (2009), 1121–1128.

Andreas Buja, Dianne Cook, and Deborah F Swayne. 1996. Interactive high-dimensional data visualization. *Journal of Computational and Graphical Statistics* 5, 1 (1996), 78–99.

Stephen M Casner. 1991. Task-analytic approach to the automated design of graphic presentations. *ACM Transactions on Graphics (TOG)* 10, 2 (1991), 111–151.

RGG Cattell. 1980. An entity-based database user interface. In *Proceedings of the 1980 ACM SIGMOD international conference on Management of data*. ACM, 144–150.

Sye-Min Chan, Ling Xiao, John Gerth, and Pat Hanrahan. 2008. Maintaining interactivity while exploring massive time series. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology, IEEE VAST 2008, Columbus, Ohio, USA, 19-24 October 2008*. IEEE Computer Society, 59–66. DOI:http://dx.doi.org/10.1109/VAST.2008.4677357

Surajit Chaudhuri, Gautam Das, and Vivek R. Narasayya. 2007. Optimized stratified sampling for approximate query processing. *ACM Trans. Database Syst.* 32, 2 (2007), 9. DOI:http://dx.doi.org/10.1145/1242524.1242526

Surajit Chaudhuri, Rajeev Motwani, and Vivek R. Narasayya. 1999. On Random Sampling over Joins. In *Proceedings SIGMOD*. 263–274.

Mei C Chuah and Steven F Roth. 2003. Visualizing common ground. In *Information Visualization, 2003. IV 2003. Proceedings. Seventh International Conference on*. IEEE, 365–372.

Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh (Eds.). 1999. *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*. ACM Press.

Glynn Dennis Jr, Brad T Sherman, Douglas A Hosack, Jun Yang, Wei Gao, H Clifford Lane, Richard A Lempicki, and others. 2003. DAVID: database for annotation, visualization, and integrated discovery. *Genome biol* 4, 5 (2003), P3.

Mark Derthick, John Kolojejchick, and Steven F Roth. 1997. An interactive visual query environment for exploring data. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*. ACM, 189–198.

Punit R. Doshi, Elke A. Rundensteiner, and Matthew O. Ward. 2003. Prefetching for Visual Data Exploratio. In *Eighth International Conference on Database Systems for Advanced Applications (DASFAA '03), March 26-28, 2003, Kyoto, Japan*. IEEE Computer Society, 195–202. DOI:http://dx.doi.org/10.1109/DASFAA.2003.1192383

Stephen G Eick, M Andrew Eick, Jesse Fugitt, Brian Horst, Maxim Khailo, and Russell A Lankenau. 2007. Thin client visualization. In *Visual Analytics Science and Technology, 2007. VAST 2007. IEEE Symposium on*. IEEE, 51–58.

Niklas Elmqvist and Jean-Daniel Fekete. 2010. Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines. *IEEE Trans. Vis. Comput. Graph.* 16, 3 (2010), 439–454. DOI:http://dx.doi.org/10.1109/TVCG.2009.84

Danyel Fisher. 2007. Hotmap: Looking at Geographic Attention. *IEEE Trans. Vis. Comput. Graph.* 13, 6 (2007), 1184–1191. DOI:http://dx.doi.org/10.1109/TVCG.2007.70561

Danyel Fisher. 2011. Incremental, approximate database queries and uncertainty for exploratory visualization. In *IEEE Symposium on Large Data Analysis and Visualization, LDAV 2011, Providence, Rhode Island, USA, 23-24 October, 2011*, David Rogers and Cláudio T. Silva (Eds.). IEEE, 73–80. DOI:http://dx.doi.org/10.1109/LDAV.2011.6092320

Danyel Fisher, Steven M. Drucker, and Arnd Christian König. 2012a. Exploratory Visualization Involving Incremental, Approximate Database Queries and Uncertainty. *IEEE Computer Graphics and Applications* 32, 4 (2012), 55–62. DOI:http://dx.doi.org/10.1109/MCG.2012.48

Danyel Fisher, Igor O. Popov, Steven M. Drucker, and m. c. schraefel. 2012b. Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster. In *CHI Conference on Human Factors in Computing Systems, CHI '12, Austin, TX, USA - May 05 - 10, 2012*, Joseph A. Konstan, Ed H. Chi, and Kristina Höök (Eds.). ACM, 1673–1682. DOI:http://dx.doi.org/10.1145/2207676.2208294

Jade Goldstein, Steven F Roth, John Kolojejchick, and Joe Mattis. 1994. A framework for knowledge-based interactive data exploration. *Journal of Visual Languages & Computing* 5, 4 (1994), 339–363.

Jim Gray, Adam Bosworth, Andrew Layman, and Hamid Pirahesh. 1996. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. In *Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana*, Stanley Y. W. Su (Ed.). IEEE Computer Society, 152–159. DOI:http://dx.doi.org/10.1109/ICDE.1996.492099

Jarek Gryz, Junjie Guo, Linqi Liu, and Calisto Zuzarte. 2004. Query Sampling in DB2 Universal Database. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, Gerhard Weikum, Arnd Christian König, and Stefan Deßloch (Eds.). ACM, 839–843. DOI:http://dx.doi.org/10.1145/1007568.1007664

Peter J. Haas and Joseph M. Hellerstein. 1999. Ripple Joins for Online Aggregation, See Delis et al. [1999], 287–298. DOI:http://dx.doi.org/10.1145/304182.304208

Jeffrey Heer and Sean Kandel. 2012. Interactive analysis of big data. *ACM Crossroads* 19, 1 (2012), 50–54. DOI:http://dx.doi.org/10.1145/2331042.2331058

Jeffrey Heer and Ben Shneiderman. 2012. Interactive dynamics for visual analysis. *Commun. ACM* 55, 4 (2012), 45–54. DOI:http://dx.doi.org/10.1145/2133806.2133821

Joseph M. Hellerstein, Ron Avnur, Andy Chou, Christian Hidber, Chris Olston, Vijayshankar Raman, Tali Roth, and Peter J. Haas. 1999. Interactive Data Analysis: The Control Project. *IEEE Computer* 32, 8 (1999), 51–59. DOI:http://dx.doi.org/10.1109/2.781635

Matthias Hemmje, Clemens Kunkel, and Alexander Willett. 1994. LyberWorlda visualization user interface supporting fulltext retrieval. In *SIGIR94*. Springer, 249–259.

Robert J Hendley, Nick S Drew, Andrew M Wood, and Russell Beale. 1995. Case study. Narcissus: visualising information. In *Information Visualization, 1995. Proceedings.* IEEE, 90–96.

Christopher F Herot. 1980. Spatial management of data. *ACM Transactions on Database Systems (TODS)* 5, 4 (1980), 493–513.

W Hibbard, J Kellum, and B Paul. 1990. Vis5D Version 5.2. *Visualization Project, University of Wisconsin–Madison Space Science and Engineering Center* (1990).

Xiaohua Hu, Tsau Young Lin, Vijay Raghavan, Benjamin W. Wah, Ricardo A. Baeza-Yates, Geoffrey Fox, Cyrus Shahabi, Matthew Smith, Qiang Yang, Rayid Ghani, Wei Fan, Ronny Lempel, and Raghunath Nambiar (Eds.). 2013. *Proceedings of the 2013 IEEE International Conference on Big Data, 6-9 Octo-*

ber 2013, Santa Clara, CA, USA. IEEE. http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6679357

Jean-Francois Im, Felix Giguere Villegas, and Michael J. McGuffin. 2013. VisReduce: Fast and responsive incremental information visualization of large datasets, See Hu et al. [2013], 25–32. DOI:http://dx.doi.org/10.1109/BigData.2013.6691710

Allan S Jacobson, Andrew L Berkin, and Martin N Orton. 1994. LinkWinds: interactive scientific data analysis and visualization. *Commun. ACM* 37, 4 (1994), 42–52.

Chris Jermaine, Alin Dobra, Subramanian Arumugam, Shantanu Joshi, and Abhijit Pol. 2006. The Sort-Merge-Shrink join. *ACM Trans. Database Syst.* 31, 4 (2006), 1382–1416. DOI:http://dx.doi.org/10.1145/1189775

Uwe Jugel, Zbigniew Jerzak, Gregor Hackenbroich, and Volker Markl. 2014. Faster Visual Analytics through Pixel-Perfect Aggregation. *Proceedings of the VLDB Endowment* 7, 13 (2014).

Niranjan Kamat, Prasanth Jayachandran, Karthik Tunga, and Arnab Nandi. 2014. Distributed and interactive cube exploration. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, Isabel F. Cruz, Elena Ferrari, Yufei Tao, Elisa Bertino, and Goce Trajcevski (Eds.). IEEE, 472–483. DOI:http://dx.doi.org/10.1109/ICDE.2014.6816674

Daniel A Keim and H-P Kriegel. 1994. VisDB: Database exploration using multidimensional visualization. *Computer Graphics and Applications, IEEE* 14, 5 (1994), 40–49.

Konstantinos Konstantinides and John Robert Rasure. 1994. The Khoros software development environment for image and signal processing. *Image Processing, IEEE Transactions on* 3, 3 (1994), 243–252.

David Koop, Carlos E Scheidegger, Steven P Callahan, Juliana Freire, and Cláudio T Silva. 2008. Viscomplete: Automating suggestions for visualization pipelines. *Visualization and Computer Graphics, IEEE Transactions on* 14, 6 (2008), 1691–1698.

Robert Kosara, Silvia Miksch, and Helwig Hauser. 2001. Semantic Depth of Field. In *IEEE Symposium on Information Visualization 2001 (INFOVIS'01), San Diego, CA, USA, October 22-23, 2001.*, Keith Andrews, Steven F. Roth, and Pak Chung Wong (Eds.). IEEE Computer Society, 97–104. DOI:http://dx.doi.org/10.1109/INFVIS.2001.963286

Xiaolei Li, Jiawei Han, Zhijun Yin, Jae-Gil Lee, and Yizhou Sun. 2008. Sampling cube: a framework for statistical olap over sampling data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, Jason Tsong-Li Wang (Ed.). ACM, 779–790. DOI:http://dx.doi.org/10.1145/1376616.1376695

Zhicheng Liu, Biye Jiang, and Jeffrey Heer. 2013. *imMens*: Real-time Visual Querying of Big Data. *Comput. Graph. Forum* 32, 3 (2013), 421–430. DOI:http://dx.doi.org/10.1111/cgf.12129

Miron Livny, Raghu Ramakrishnan, Kevin Beyer, Guangshun Chen, Donko Donjerkovic, Shilpa Lawande, Jussi Myllymaki, and Kent Wenger. 1997a. DEVise: integrated querying and visual exploration of large datasets. In *ACM SIGMOD Record*, Vol. 26. ACM, 301–312.

Miron Livny, Raghu Ramakrishnan, Kevin S. Beyer, Guangshun Chen, Donko Donjerkovic, Shilpa Lawande, Jussi Myllymaki, and R. Kent Wenger. 1997b. DEVise: Integrated Querying and Visualization of Large Datasets. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA.*, Joan Peckham (Ed.). ACM Press, 301–312. DOI:http://dx.doi.org/10.1145/253260.253335

Jock Mackinlay. 1986. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics (TOG)* 5, 2 (1986), 110–141.

Allen D Malony, David H Hammerslag, and David J Jablonowski. 1992. *Traceview: A trace visualization tool*. Springer.

Nancy H McDonald and Michael Stonebraker. 1975. CUPID-The Friendly Query Language.. In *ACM Pacific*. 127–131.

Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, and Theo Vassilakis. 2010. Dremel: interactive analysis of web-scale datasets. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 330–339.

Charles Joseph Minard. 1812. Carte figurative des pertes successives en hommes de l'arm ee qu'Annibal conduisit d'Espagne en italie en traversant les Gaules (selon Polybe). *Carte figurative des pertes successives en hommes de l'arm ee franc aise dans la campagne de Russie* 1813 (1812).

C Olsten, Michael Stonebraker, Alexander Aiken, and Joseph M Hellerstein. 1998. VIQING: Visual interactive querying. In *Visual Languages, 1998. Proceedings. 1998 IEEE Symposium on*. IEEE, 162–169.

Chris Olston and Jock D. Mackinlay. 2002. Visualizing Data with Bounded Uncertainty. In *2002 IEEE Symposium on Information Visualization (InfoVis 2002), 27 October - 1 November 2002, Boston, MA, USA*, Pak Chung Wong and Keith Andrews (Eds.). IEEE Computer Society, 37–40. DOI:http://dx.doi.org/10.1109/INFVIS.2002.1173145

Eric D Petajan, Yves D Jean, Dan Lieuwen, and Vinod Anupam. 1997. DataSpace: An automated visualization system for large databases. In *Electronic Imaging'97*. International Society for Optics and Photonics, 89–98.

J Priestley. 1765. A chart of biography, London. *British Library, London:. I* (1765).

Ramana Rao and Stuart K Card. 1994. The table lens: merging graphical and symbolic representations in an interactive focus+ context visualization for tabular information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 318–322.

Steven F Roth, Peter Lucas, Jeffrey A Senn, Cristina C Gomberg, Michael B Burks, Philip J Stroffolino, AJ Kolojechick, and Carolyn Dunmire. 1996. Visage: a user interface environment for exploring information. In *Information Visualization'96, Proceedings IEEE Symposium on*. IEEE, 3–12.

Steven F Roth and Joe Mattis. 1991. Automating the presentation of information. In *Artificial Intelligence Applications, 1991. Proceedings., Seventh IEEE Conference on*, Vol. 1. IEEE, 90–97.

Jibonananda Sanyal, Song Zhang, Gargi Bhattacharya, Philip Amburn, and Robert J. Moorhead. 2009. A User Study to Compare Four Uncertainty Visualization Methods for 1D and 2D Datasets. *IEEE Trans. Vis. Comput. Graph.* 15, 6 (2009), 1209–1218. DOI:http://dx.doi.org/10.1109/TVCG.2009.114

Arvind Satyanarayan, Kanit Wongsuphasawat, and Jeffrey Heer. 2014. Declarative interaction design for data visualization. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 669–678.

Ben Shneiderman. 1996. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Visual Languages/Human-Centric Computing Languages and Environments*. 336–343. DOI:http://dx.doi.org/10.1109/VL.1996.545307

Ben Shneiderman. 2008. Extreme visualization: squeezing a billion records into a million pixels. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 3–12.

John Snow and BW Richardson. 1965. *Snow on Cholera: Being a Reprint of Two Papers by John Snow, MD, Together with a Biographical Memoir by BW Richardson, and an Introduction by Wade Hampton Frost, MD.* Hafner.

Michael Spenke, Christian Beilken, and Thomas Berlage. 1996. FOCUS: the interactive table for product comparison and selection. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*. ACM, 41–50.

Anselm Spoerri. 1993. InfoCrystal: A visual tool for information retrieval & management. In *Proceedings of the second international conference on Information and knowledge management*. ACM, 11–20.

Thomas C Sprenger, Markus H Gross, Daniel Bielser, and T Strasser. 1998. IVORY-An Object-Oriented Framework for Physics-Based Information Visualization in Java. In *Information Visualization, 1998. Proceedings. IEEE Symposium on*. IEEE, 79–86.

Chris Stolte, Diane Tang, and Pat Hanrahan. 2002. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *Visualization and Computer Graphics, IEEE Transactions on* 8, 1 (2002), 52–65.

Michael Stonebraker, Jolly Chen, Nobuko Nathan, Caroline Paxson, and Jiang Wu. 1993. Tioga: Providing data management support for scientific visualization applications. In *VLDB*, Vol. 93. Citeseer, 25–38.

Alexander Streit, Binh Pham, and Ross Brown. 2008. A Spreadsheet Approach to Facilitate Visualization of Uncertainty in Information. *IEEE Trans. Vis. Comput. Graph.* 14, 1 (2008), 61–72. DOI:http://dx.doi.org/10.1109/TVCG.2007.70426

Edward Tufte. 1990. *Envisioning Information*. Graphics Press, Cheshire, CT, USA.

Craig Upson, Thomas A Faulhaber Jr, David Kamins, David Laidlaw, David Schlegel, Jefrey Vroom, Robert Gurwitz, and Andries Van Dam. 1989. The application visualization system: A computational environment for scientific visualization. *Computer Graphics and Applications, IEEE* 9, 4 (1989), 30–42.

Matthew O Ward. 1994. Xmdvtool: Integrating multiple methods for visualizing multivariate data. In *Proceedings of the Conference on Visualization'94*. IEEE Computer Society Press, 326–333.

Richard Wesley, Matthew Eldridge, and Pawel T Terlecki. 2011. An analytic data engine for visualization in tableau. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 1185–1194.

Richard Michael Grantham Wesley and Pawel Terlecki. 2014. Leveraging compression in the tableau data engine. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 563–573.

Craig M. Wittenbrink, Alex Pang, and Suresh K. Lodha. 1996. Glyphs for Visualizing Uncertainty in Vector Fields. *IEEE Trans. Vis. Comput. Graph.* 2, 3 (1996), 266–279. DOI:http://dx.doi.org/10.1109/2945.537309

Harry KT Wong, Ivy Kuo, and others. 1982. GUIDE: Graphical User Interface for Database Exploration.. In *VLDB*. 22–32.

Pak Chung Wong, Beth Hetzler, Christian Posse, Mark Whiting, Susan Havre, Nick Cramer, Anuj Shah, Mudita Singhal, Alan Turner, and Jim Thomas. 2004. In-spire infovis 2004 contest entry. In *Information Visualization, IEEE Symposium on*. IEEE, r2–r2.

Allison Woodruff, Chris Olston, Alexander Aiken, Michael Chu, Vuk Ercegovac, Mark Lin, Mybrid Spalding, and Michael Stonebraker. 2001. DataSplash: A direct manipulation environment for programming semantic zoom visualizations of tabular data. *Journal of Visual Languages & Computing* 12, 5 (2001), 551–571.

Eugene Wu, Leilani Battle, and Samuel R. Madden. 2014. The Case for Data Visualization Management Systems [Vision Paper]. *Proceedings of the VLDB Endowment* (2014).

Moshé M Zloof. 1975. Query by example. In *Proceedings of the May 19-22, 1975, national computer conference and exposition*. ACM, 431–438.

Received TBA; revised TBA; accepted TBA

APPENDIX

A Selection of Domain-Specific Visualization Systems

As mentioned in Section 1, we are primarily interested in tools which are not domain-specific and aimed to visualize relational databases. However, over the years a great number of tools for data visualization was designed and many of them were intended to support different types of scientific research. We list them here for completeness.

(1) DAVID [Dennis Jr et al. 2003] - a web-accessible program integrating functional genomic annotations with intuitive graphical summaries
(2) IVORY - a platform-independent framework for visualization [Sprenger et al. 1998] in physics
(3) SDMS [Herot 1980] - a spatial data management system which presented the geography and weather prediction information
(4) Vis5D [Hibbard et al. 1990] - a system for interactive visualization of data sets produced by numerical weather prediction
(5) GeoBoost [Eick et al. 2007] - a thin client visualization framework which focuses on geospatial visualization and uses Scalable Vector Graphics
(6) Lyberworld [Hemmje et al. 1994] - a visualization interface supporting full-text retrievalm and IN-SPIRE [Wong et al. 2004] - designed for visualizing document collections
(7) AVS Explorer [Upson et al. 1989] - a system for developing interactive scientific visualization applications with a minimum of programming effort
(8) ParaGraph [Bailey et al. 1990] - a graph editor supporting parallel programming environments
(9) LinkWinds [Jacobson et al. 1994] - an interactive scientific data analysis and visualization system applying a data-linking paradigm resulting in a system which functions much like a graphical spreadsheet
(10) VisPortal [Bethel et al. 2003] - a system for grid-based visualization services and focused on distributed visualization algorithms).
(11) GUIDE [Wong et al. 1982] - a graphical user interface for database exploration which offered a graphics interface to the user used to present a database schema in a for of a network of entities and relationships where queries were formulated and represented graphically
(12) E-R [Cattell 1980] - a user interface to a data base designed for casual interactive use in which data displayed to the user was based upon entities participating in relationships, rather than upon relations alone as in the relational data model
(13) QBE (Query by Example) [Zloof 1975] - designed for users having no computer or mathematical background.

Other tools which did not fit into our survey due to the fact that they were not designed for visualization of data bases but we considered worth of mentioning are:

(14) CUPID [McDonald and Stonebraker 1975] - a friendly query language designed to support non-programmer integration with relational databases
(15) Traceview [Malony et al. 1992] - a system for visualization and trace files manipulation
(16) Khoros [Konstantinides and Rasure 1994] - a data flow visual language allowing to create block diagrams integrating software development environment for information processing and visualization
(17) XmdvTool [Ward 1994] - a software package for interactive visual exploration of multivariate data sets
(18) Table Lens [Rao and Card 1994] and FOCUS [Spenke et al. 1996] - visualization systems providing table displays that present data in a relational table view, using simple graphics in the cells to communicate quantitative values
(19) Narcissus [Hendley et al. 1995] - a tool for visualization leading users to form an intuitive understanding of the structure and behaviour of their domain allowing them to manipulate objects within their system
(20) VR-VIBE [Benford et al. 1995] - a virtual reality application intended to support the co-operative analysis of document stores
(21) XGobi [Buja et al. 1996] - a tool providing predefined visualizations intended for exploring high multidimensional data.
(22) ILOG Discovery [Baudel 2004] - a program designed as an information visualization editor allowing browsing the visualization design space of data sets
(23) VisComplete [Koop et al. 2008] - a tool for computing correspondences between existing pipeline subgraphs from the database, and use these to predict sets of likely pipeline additions to a given partial pipeline. By presenting these sub-graphs in an interface users could use suggested completions when creating visualizations
(24) Protovis [Bostock and Heer 2009] - an embedded domain-specific language for constructing visualizations by composing simple graphical marks such as bars, lines and labels
(25) Quadrigram [Satyanarayan et al. 2014] - a data visualization web-based service launched in 2012 and based on a visual programming language.