Interactive Visualization of Large Data Sets

Parke Godfrey, Jarek Gryz, and Piotr Lasek

Abstract—Visualization provides a powerful means for data analysis. But to be practical, visual analytics tools must support smooth and flexible use of visualizations at a fast rate. This becomes increasingly onerous with the ever-increasing size of real-world datasets. First, large databases make interaction more difficult once query response time exceeds several seconds. Second, any attempt to show all data points will overload the visualization, resulting in chaos that will only confuse the user. Over the last few years, substantial effort has been put into addressing both of these issues and many innovative solutions have been proposed. Indeed, *data visualization* is a topic that is too large to be addressed in a single survey paper. Thus, we restrict our attention here to *interactive* visualization of *large* data sets. Our focus then is skewed in a natural way towards query processing problem—provided by an underlying database system—rather than to the actual data visualization problem.

Index Terms—Database visualization, data aggregation, data indexing, interactive visualization, big data

1 Introduction

One picture is worth a thousand words. The idea of replacing a complex narrative with a single image may have become a cliché 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. Indeed, it has been suggested recently [1] that for extreme scale data sets, scientists might be better off analysing images *instead* of actual data.

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 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 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 [2]. 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 (at least, for today) 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

- *P. Godfrey and J. Gryz are with the Department of Electrical Engineering and Computer Science, York University, Toronto, ON, Canada. E-mail: {godfrey, jarek}@cse.yorku.ca.*
- *P. Lasek is with York University, Canada, and Rzeszów University, Poland. E-mail: plasek@cse.yorku.ca.*

Manuscript received 21 Sept. 2015; revised 14 Apr. 2016; accepted 15 Apr. 2016. Date of publication 21 Apr. 2016; date of current version 5 July 2016. Recommended for acceptance by A. Aboulnaga.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2016.2557324

discuss here the challenge of visualizing data sets of high dimensionality. This is another meaning of "large" pointed out in [3]. 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 current work in visualization that does not explicitly address the issue of data processing scalability.

The paper is organized as follows. In Section 2, we provide a background of how the concept of 'interactive visualization' has developed over the last 30 years. Our intention is not to give a historical overview of the area (given the sheer amount of work done here this would be impossible in a short paper), but rather to point out the ideas and solutions that are still used in current systems. The visual summary of the reviewed systems is shown in Fig. 6 and the inherited principles and standards recapped in Section 2.4.

Section 3 provides and overview of systems and solutions that address the *data processing scalability* in large scale visualization. We organize the section by dividing research in this area into four major domains along two orthogonal dimensions. The first dimension refers to the *type of data* against which queries are executed, that is, whether the data is preprocessed (by pre-computing queries and storing their answers) or not. The second dimension refers to the *type of answers* expected, that is, whether they are supposed to be exact or just approximate (such as provided by sampling).

In Section 4, we discuss the problem of *visual scalablility*. We review the solutions used at the data processing stage (via filtering, aggregation, or sampling) or at the data visualization stage (via binned aggregation). We also point out how human perception factors should be accommodated in the visualization process.

In Section 5 we attempt to contrast the expectations the users have of interactive visualization and the performance that realistic systems can provide. Our reality check is that the community must re-calibrate its expectations in specific ways so that the expectations are achievable. Our message is not all bad news, however. Many systems discussed in this review deliver excellent performance by making restrictions that are perfectly acceptable for their respective domains and applications. Still, 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. The intent of this paper is to be a step in this direction.

2 The Origins of Interactive Data Visualization

Over the last three hundred years inspiring data visualizations have been created. The famous Chart of Biography by Joseph Priestley from 1765 [4], Napoleon's Russian Campaign of 1812 [5], and the map of Cholera Clusters in London in 1854 [6] which helped to identify the sources of water contamination are three such. These and other historically important visualizations [7] 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 and 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 [8], Bertin addresses different issues related to the process of creating good visualizations. He argues 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 modifying the presentation. He should also be 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 detail 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 such as charts, diagrams, networks, maps or symbols. Bertin's work was the first attempt to provide theoretical foundations of 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.

2.1 Early Systems

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 [9] 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 Bertin's conceptual framework to work with data stored in a relational database was developed in 1986 [10]. The goal of the tool called APT (A Presentation Tool) was to develop an application independent system which would be capable of automatically creating visual representations of relational data . In order to achieve good results, the authors of APT codified graphic design elements and made the assumption

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

```
rel = x->y &
     ~Numeric(x) &
     Numeric(y) &
     Cardinality(x) < 20 &
     LineObjs(barchart,lines) &
     VertAxis(barchart,vaxis) &
     Length(lines, len, vaxis) &
     Encodes(len, rel(x)) & ...
     =>
     Presents(barchart, rel)
```

Fig. 1. A sample rule designed to generate a bar-chart in APT [10]. First two lines are expressiveness and effectiveness conditions, respectively. The number of bars in a new-created visualization is limited by the effectiveness condition since too many bars may be difficult to analyze.

that graphical presentations (visualizations) are sentences in a graphical language (Fig. 1). 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 [11]. This tool employed a task-analytic approach which meant that users could specify a logical description of a visualization task to be performed (Fig. 2). 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.

2.2 The Golden Decade

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. Mackinlay, for example, considered animation and 3-D presentation as means which could be used in the process of data visualization [10]. What is even more interesting, perhaps, is that researchers noticed the need to design the systems that were *interactive*. Nevertheless, the interactivity in the early 80s and 90s was only considered 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*... [12] was implemented in most of the systems 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.

```
(cost flight117 179 12:50)->
         (horzpos flight117 9.83)
(cost flight117 179) ->
         (height flight117 1.79)
(avail. flight239 ok) ->
         (shading fligh239 white)
```

Fig. 2. Translation of logical facts into so-named graphical facts in the BOZ system [11].

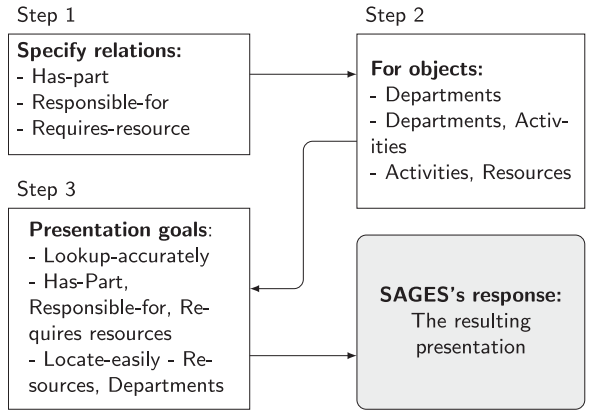


Fig. 3. A sample SAGE's initial request required for creating the resulting response [13].

For example, in the case of the SAGE system [13], 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) (Fig. 3).

Other systems based on SAGE used similar approaches of semi-interactive exploration of databases. IDES [14], 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 re-issue 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 repeat 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 [15] 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 new challenges: how to deal with data that do not have natural representation as a map; how to fit large data into small

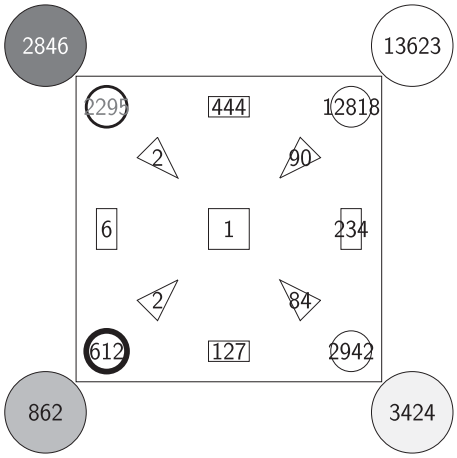


Fig. 4. A picture presenting the InfoCrystal's graphical language [16].

screen; and how to find the best highlighting methods such as points, colours, flicker, and light.

Another step towards more interactive visualization and exploration was the idea of using stored results of visualization in a form of a *slide show* (Visage [18]). 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 in Visage led to creation of Visual Query Language (VQE) [19] which added capabilities of direct manipulation and exploration of databases. 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 to support building queries by means of a specialized graphical language. Such a language was used in InfoCrystal [16]. In this case, however, a graphical language was used both for defining queries and visualizing results. In Fig. 4 we illustrate that graphical language. The number associated with an icon indicates how many of the retrieved documents satisfy the conditions represented by it. The icons with rank 2 that lie diagonally opposite each other are represented twice, because the ranking principle takes precedence over the proximity principle. Its structure is composed of so-named *crystals* based on Venn's diagrams [20]. The elements of graphical queries can 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 screen's 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 [21] (a product enabling data sharing, visualization and messaging) and later into Command Post of the Future (CPOF) [21] a software allowing military commanders to manage a battlefield.

Later systems paid more attention to design flexible graphical user interface so that the users could perform a number of visual operations such as zooming, 3D manipulation, panning, filtering and selection of details. Those systems were also *interactive* thanks to numerous sliders (similarly to [17]). For example VIS [22] and IVEE [17] (Fig. 5) and eventually

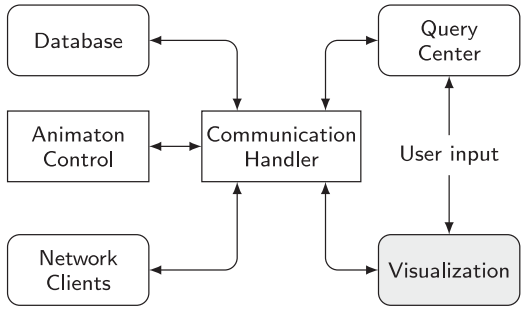


Fig. 5. A schema of the IVEE system [17].

Spotfire [23] were designed to use the concepts of dynamic query filters (allowing users to 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 [24]). Later, a concept of dynamic queries [25] 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 data 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 expensive computations. Nevertheless, the necessity of using approximations was prompted not by the size of a dataset but rather by limitations 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 whose result could be used to refine it by properly adjusting the widgets. So the process of query refinement was not optimized, did not use cached results and in order to get answers the data had to be retrieved again. As regards 3D manipulation mentioned above, the DataSpace [26] system needs some attention. It was a mouse-based 3D 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 [15] 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 that the number of visualised dimensions could be 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

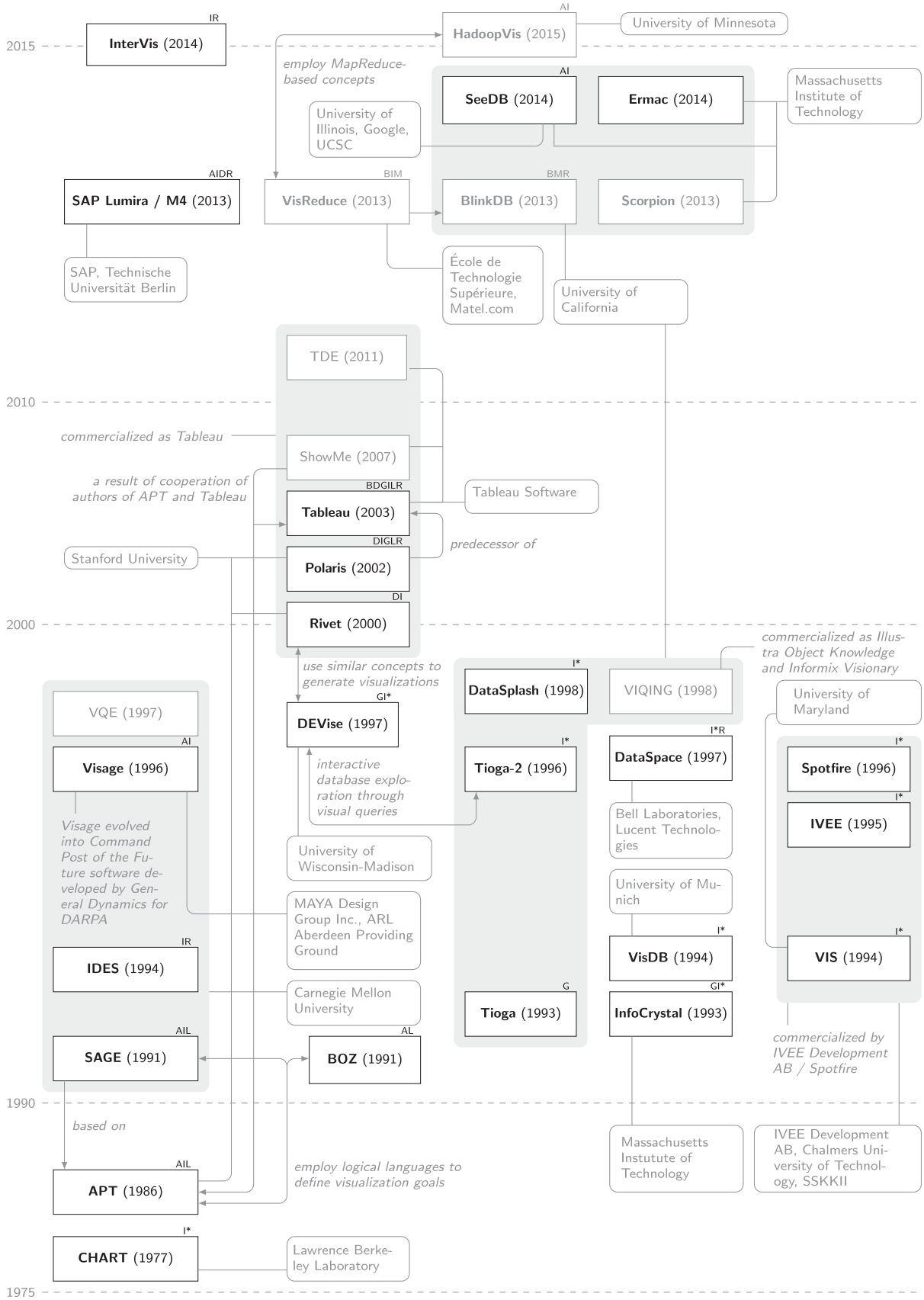


Fig. 6. A map of database visualization tools covering the years 1977 to 2015. A—generates visualizations automatically, B—operates on Big Data, D—uses an integrated database or a table-like structures to store imported data, G—uses a graphical language to define visualizations, I—supports interaction (*—naively, e.g., by reloading data based on a new query), L—uses a special language to define visualizations, M—uses a concept of Map-Reduce to deal with big data, and R—has a function to aggregate data. Gray area denotes a family of systems developed by the same team; system name in a gray rectangle denotes a system which does not fulfill our database visualization tool criteria, however, it was important in terms of implemented methods; names of universities or companies are provided in rounded gray rectangles.

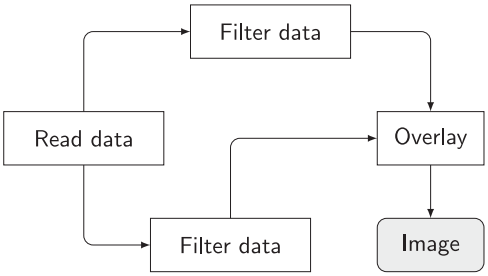


Fig. 7. Tioga's sample receipe [27].

some systems were using sliders, buttons, text boxes and other widgets, Tioga [27] and Tioga-2 [28] introduced *recipes* as definitions of a whole process of data visualization. A recipe (Fig. 7) 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 to help a user 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 the 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 [29]. This family of systems used a special graphical environment for defining queries called VIQING [30], 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 [31] 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).

2.3 Towards Modern Visualization Systems

Tools designed in early 2000s put even more emphasis on *interactive* visual data analysis. For example, in case of Rivet [32] 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 (Fig. 8) 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. In terms of design, Rivet used a homogeneous data model and separation of data objects from visual objects which allowed creating one visualization having multiple views of the same data. Rivet also used a mechanisms of selectors (objects identifying data subsets) and visual *metaphors* (functions translating data objects into their graphical representations). Expertise gained during development of Rivet led to formulation of VisQL—a language for the Polaris system [33] 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.[2] Soon after Polaris was created, Tableau Software was founded and used Polaris' source code to create the a first product of the company.

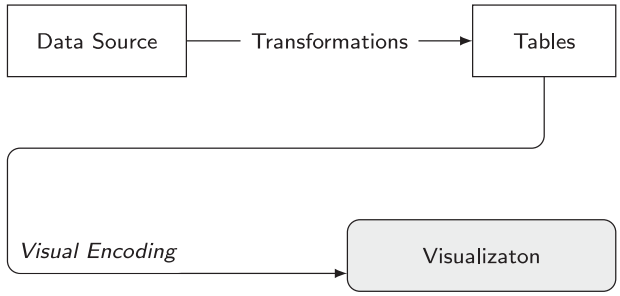


Fig. 8. A simplified diagram of information flow in the Rivet system [32].

2.4 The Legacy

In eighties Mackinlay formulated [10] two main assumptions that a visualization system should satisfy, namely *expressiveness* and *efficiency*. Nearly at the same time it became clear that visualization systems should also be *interactive* [11]. And until today, those assumptions are often referred to when the design decisions about the architecture of a new visualization system are made. The early systems addressed the issue of expressiveness (i.e., an ability to express a set of facts by means of a visualization) and effectiveness (i.e., an ability of a user to understand a graphical representation of data) by:

- Employing Bertin's idea of mapping data objects to their visual representations (e.g., points for representing location, lines for length) but also matching types of charts with types of data to be presented (e.g., graphs for representing networks, scatter plots for representing positions, etc.).
- Introducing new functions of a user's interface such as zooming in/out, panning, filtering, brushing, selection of details, etc. (VIS, IVEE, Spotfire, Polaris, Tableau).
- Trying to provide a user with functions allowing to create visualizations *automatically* based on a codified set of *good* design principles (e.g., APT, SAGE).
- Using a special *graphical language* to define visualizations (or presentation goals) (BOZ, SAGE, InfoCrystal, Visage, Polaris, Tableau).[3]

2. Some other extensions to VizQL (e.g., related to data cleaning) have been recently proposed in [34].

3. Recent examples of similar approach for defining visualizations are: Protovis [35], its successor Ractive Vega [36] or D^3 [37]) designed for interactive exploration of stream data.

		Exact answers	Approximate answers	
Raw data		**Dremel** [38]	-	**Control** [45], **sampleAction** [46, 47], **VisReduce** [48], [49]
Pre-processed data	Static	**imMens** [44]	[50], [51], **AQUA** [52]	**BlinkDb** [53]
	Dynamic	**XmdvTool** [54], **ATLAS** [55]	[56]	**DICE** [57]

Fig. 9. Categorization of data processing paradigms with some representative systems. Visualization systems are underlined.

- Giving a user means to recreate visualization by adjusting query parameters with using user interface elements such as sliders, text boxes and buttons, in other words, by making visualizations *interactive*.

Even though the data sets were not large at the time (in today's sense) the interactivity was considered as an important feature of a visualization system. Efficient redrawing of a visualization based on adjusted values of parameters allowed users to easily create more meaningful graphical representations of data. Due to usually small data set sizes, systems that supported interactivity (CHART, APT, SAGE, InfoCrystal, IDES, VisDB, VIS, IVEE, Visage, Tioga-2, Spotfire, DataSpace, DEVise, DataSplash, Rivet, Polaris, Tableau) worked in a simple way (at least in their early version): Each time parameters of a visualization (or query) changed, a new portion of data was loaded; then the visualization was refreshed and if a user was not satisfied with the result, these steps could be repeated. In some cases however, certain ideas were implemented to improve systems' efficiency of accessing data, for example:

- Integrated databases or a table-like structures were used to store imported data to speed up data processing for visualization (Rivet, Polaris, Tableau).
- Aggregation, in case of larger data sets, helped to address the problem of how to *squeeze*a sheer amount of graphical objects into a small (in terms of a number of pixels) screens and ensure good responsiveness of the systems (IDES, DataSpace, Polaris, Tableau).
- Filtering was applied to decrease amount of data to be visualized like in VIS, IVEE, Spotfire or in VisDB for filtering attributes by their relevancy to a query.

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 Fig. 9. 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 Fig. 9). 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 Fig. 9.)

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. An obvious way of improving performance of any system is to throw more hardware at it. For example, Google's *Big Query* (a commercial, cloud-based version of the *Dremel* [38] 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. *Big Query* puts together a number of well-known techniques: parallelism on a shared cluster of commodity machines, columnar storage, and data compression to process terabytes of data in a matter of seconds. Similarly, the *MapD (Massively Parallel Database)* project at MIT [39] takes advantage of the immense computational power available in commodity-level, of-the-shelf GPUs and sub-USD1000 desktops to build

clusters of thousands of nodes.[4] It is unlikely, however, that systems like *MapD* or *Big Query* will find their way into a wider market: the cost of the system (thousands of machines in a cluster), and a proprietary query language and query processing scheme put it out of reach but for the largest enterprises.

Another recent trend in high-performance systems calls for the use of main-memory databases. *HyPer* [41] combines main-memory capacity of several TB with vast multi-core parallel processing power. To unleash unprecedented computing power of such systems a radical reengineering of database technology was necessary: low-overhead data representation, compiled instead of interpreted code, high-performance indexing and low-cost transaction synchronization. Systems like *HyPer* are rather expensive when TB of data are stored and processed, but are likely to be affordable within years.

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 [42] 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 materialized views have been called 'cubes' [43]. 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 [44].

The advantages of data pre-processing are as obvious as the limitations: queries can be answered fast as long as *their* answers have been previously stored.[5] 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 ad hoc queries, static pre-processing is of limited use.

Not surprisingly, few systems (the *Hotmap* project [58] 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 [54]. 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 these queries. The *XmdvTool* [54] 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 [55] 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 the 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 [56].

4. Extremely high-performance systems similar to *MapD* include *EXASolution*, Amazon's cloud-based *Redshift* [40] and *Snowflake*.

5. 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.

Example. 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. Most of the joins executed in real-world databases are between primary and foreign key attributes. Sampling performed on the table with the foreign key is commutative with such joins.

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 they 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 *Control* project [45] uses the first specialized techniques for joins over samples, called ripple join algorithms, introduced [59]. 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 [49]. *sampleAction* system [46], [47] builds upon the ideas of the *Control* project, but here a user is also provided with a confidence bound (error measure) for the returned results.

The most straightforward approach to incremental visualization has been implemented in *VisReduce* [48]. The system is in fact similar to *Dremel* described above: it uses columnal 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.

The *DICE* system [57] combines sampling with speculative query execution (similar to the *XmdvTool*) to achieve interactive latencies for data cube exploration. The key observation made in the paper is that typical interaction with a database consists of a session of several ad-hoc queries interspersed with idle time when a user inspects the results. The solution proposed in the paper is to utilize this waiting time to speculate, execute, and cache the most likely follow-up queries. To achieve interactive latencies, *DICE* employs a novel architecture of executing queries piecemeal over individual table shards and then assembling them in a post-processing step.

3.2.3 Static Approximate Answers

To address the problem of non-commutativity of sampling with some of the 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 [51], [52], [60], [61]. A recent system in this area, *BlinkDB* [53], provides an adaptive framework that builds and maintains a set of multi-dimensional stratified samples from original data and then uses a dynamic sample selection strategy to answer queries based on the required error/response time constraint. These samples are designed to be used with specific queries so that the queries executed over them are guaranteed to return answers of (almost) 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 (*BlinkDB*, for example, assumes that the sets of columns used in queries are stable over time).

We should also note that since sample synopses are preprocessed 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 [53].

The idea of materializing samples can be pushed even further to build the entire database out of them. This idea has been explored in [50].

3.3 Tightly-Coupled and Dedicated Systems

Most visualization 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.[6]

Probably, the first attempt to build a tightly-coupled database/visualization system was the *DEVise* system [62] (Fig. 10). 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. In

6. The distinction between coupled and decoupled systems adds yet another dimension to the categorization in Fig. 9. 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.

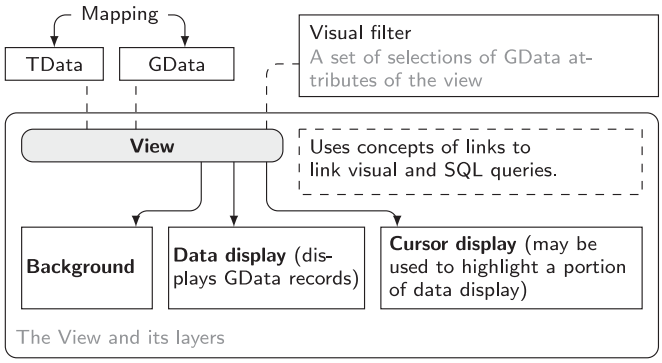


Fig. 10. A schema of the DEVise system [62].

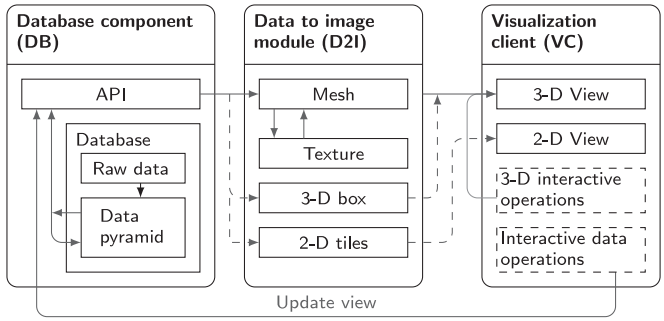


Fig. 11. A sample diagram of a tightly-coupled system.

Fig. 11 we plotted a sample diagram of a tightly-coupled system. In this case the system comprises three separate components: the Database Module (DB), the Data-to-Image module (D2I) and the Visualization Client (VC). Such a separation provides useful flexibility. The crucial elements of the system are tightly coupled with the database engine but each component can be implemented as a separate service and deployed on a different machine.

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 [54] and [55] 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 [63] 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 [64].

Recently, a call for even tighter integration of a database and visualization systems has been made [65]. 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).

Integrating a database system with visualization tools has been receiving a lot of attention recently, but there is still one element in the visualization process that has been largely overlooked: the human in the loop. Human perception has its limitations and these limitations should be recognized and utilized to minimize data processing and rendering costs. If a user cannot discriminate between the height (or color) of two different bars in a chart, then the difference in values they represent is a perfectly acceptable error in an approximation the system will provide. The key challenge is to push this filtering process into a query plan so that an aggressive optimization (for example, via an appropriate rate of sampling) can be performed. The *InterVis* system [66] provides a framework of how this can be done.

The *BigDAWG* system [67] is the most recent example of a dedicated and comprehensive system for interactive applications. The system is based on the observation that specific applications require specialized database engines ("one size does not fit all"). Thus, *BigDAWG* supports many different data models and query languages through a *polystore* system of multiple, disjoint database engines under one interface. Consider how notes about patients should be stored. The patient structured data (e.g., age, visits) should be stored in a relational database like PostgreSQL. On the other hand, when a physician wants to find notes that mention a specific illness, a text processing engine is required. A waveform data from medical devices may be best processed efficiently by an array database. For interactive applications, selecting the right database is critical for ensuring that query response times are sufficiently fast.

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 [68]. Still, further growth of sizes required further improvements of TDE's efficiency by leveraging compression techniques [69] so that Tableau products could be more interactive. As a result Tableau's in-memory database (also implemented in Vertica and PowerPivot[7]) scales up to interactive queries across millions of rows (to be fair, many commercial databases are no larger than that).

7. The origins of column store are, of course, much earlier [70], [71].

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 [72].

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 2D 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 [64].

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 a data set of an arbitrary size.

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 [73] provides a comprehensive overview of rendering techniques for displaying aggregated data.

Given a fixed number of pixels available for visualization, aggregation should be performed at a particular level given the size of the underlying data set. For example, to display the number and location of a product sold across the world, we may have to group the sales by country. However, if we want to show the same data for the US only, we may afford a more detailed aggregation at a level of individual states. Such combination of filtering and aggregation leads in a natural way to *hierarchical aggregation* where aggregates are precomputed and different levels of abstraction. This idea, explored before in the context of data cubes, has recently been implemented in data visualization as a *nanocube* [74] and *data pyramid* [75].

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 [76]: 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 [77], [78], [79], [80], [81] for more discussion.

Data reduction can also be performed *after* the data is sent from a database to the rendering tool, but *before* it is actually displayed. Some of the reduction methods employed at this stage are similar to the ones used by a database. For example, one can sample the data so that only a subset of the data points is visualized [82]. This method can be effective only if non-uniform methods are used to outweigh unusual values, but even then it can miss outliers.

When aggregation is performed by a database systems, it is much constrained by the semantics of SQL. To provide more flexibility how data can be aggregated and then displayed, more sophisticated methods have been proposed on the data visualization side. The most meaningful way of summarizing the data is via clustering [83]. Data points are aggregated based on some proximity measures with each cluster depicted using a small number of features, for example, the mean value for each dimension, the extent within each cluster for each dimension, and the cluster population. Clustering algorithm can then be applied recursively to create a hierarchy of clusters. Clustering is computationally expensive, however, and cannot be a part of an interactive process for large data sets.

Recently a new aggregation method, called *binned aggregation* has been proposed [44], [84]. Binning condenses large data set into a smaller set of binned summaries. Once each of the n original data points has been placed into one of m integer bins ($m \ll n$), the next step is to collapse the points in each bin into small number of summary statistics. Picking useful summary statistics requires balancing between computational efficiency and statistical robustness. One additional step that can be performed on data aggregated in this way is smoothing. Smoothing allows us to resolve problems with excessive variability in the summaries. This variability can arise because the bin size is too small, or because there are unusual values in the bin. A number of smoothing methods, from fast and crude to sophisticated and slower, has been proposed in [84]. Binned aggregation is able to convey both global patterns (e.g., densities) and local features (e.g., outliers) while enabling multiple levels of resolution via the bin size. It balances computational and statistical concerns without sacrificing the fidelity of the underlying data.

Data presentation is not only about efficiency in terms of time and memory needed to produce a visualization. When dealing with approximate—filtered, sampled or especially aggregated data—it is also important to generate visualizations in a scalable way. This means that for given two *correlated* queries visualizations generated should be *similar*. This similarity (or difference) can be can modelled using Weber's law.[8] Rensink and Baldride [85] as well as Harrison et al. [86] have shown that Weber's law can be used for modelling

8. Weber's or Weber-Fechner law was first expressed by Weber and formulated mathematically by Fechner. The law quantifies relation between perceptions of change in a given stimulus and states that the size of *noticable* difference is proportional the the original value of stimulus.

viewers perception can be used to evaluate simple scatterplots and other more complex types of charts respectively.

5 Conclusions and Discussion

As we mentioned in the Introduction, 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" [87]. In other words, for data exploration to be truly interactive, queries need to be responded to within a latency bound of 1-5 seconds [57]. If one puts a bar too high, however, this expectation will never be met. Let us assume that we want to build a system that supports ad-hoc SQL queries issued to a database of 1TB over a typical schema. Assume also that this has to be done within a sensible budget of, say, less than $10k. Although these assumptions seem reasonable, we must report that we have not encountered any system that would deliver truly interactive performance under these requirements. 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 [45]).

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 above. 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. Indeed, the leading commercial visualization companies, such as Tableau, support only visualization of such "small" databases.
- 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 general purpose visualization systems as it provides truly interactive performance. Although it puts some limit on the types of queries that can be answered via samples, it is a method of choice in systems where only approximate answers are expected.
- Data is pre-processed (for example, by storing materialized views). This is a method routinely used in OLAP. Although this approach does not allow for truly ad-hoc queries, it is perfectly acceptable for the majority of business applications.
- Massively parallel systems such as *Dremel*. This is the only approach that works for ad hoc queries and data sets in excess of 1TB. If one were to use it as an in-house solution, it would be too costly (3,000 machines in a cluster were used to build Dremel) and could only be used by the largest enterprises. But the recent transformation of *Dremel* into Google's cloud platform *BigQuery* [88]) opened it up to all users. Perhaps this is the way of the future for interactive visualization of large data sets.

Appendix

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) QBE (Query by Example) [89]—designed for users having no computer or mathematical background.
2) SDMS [90]—a spatial data management system which presented the geography and weather prediction information.
3) E-R [91]—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.
4) GUIDE [92]—a graphical user interface for database exploration which offered a graphics interface to the user. Database schema was presented in a form of a network of entities and relationships where queries were formulated and represented graphically.
5) AVS Explorer [93]—a system for developing interactive scientific visualization applications with a minimum of programming effort.
6) ParaGraph [94]—a graph editor supporting parallel programming environments.
7) Vis5D [95]—a system for interactive visualization of data sets produced by numerical weather prediction.
8) Lyberworld [96]—a visualization interface supporting full-text retrieval and IN-SPIRE [97]—designed for visualizing document collections.
9) LinkWinds [98]—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) IVORY—a platform-independent framework for visualization [99] in physics.
11) DAVID [100]—a web-accessible program integrating functional genomic annotations with intuitive graphical summaries.
12) VisPortal [101]—a system for grid-based visualization services and focused on distributed visualization algorithms).
13) GeoBoost [102]—a thin client visualization framework which focuses on geospatial visualization and uses Scalable Vector Graphics.
14) Profiler [103]—an integrated statistical analysis and visualization tool for data quality assessment. It applies data mining methods to automatically flag problematic data and suggests visualizations for assesing data in context.
15) HadoopVis [104]—an extensible MapReduce-based system designed for visualizing big spatial data. It provides an interface for visualizing different types of data using abstract functions without a need of going into the details of the MapReduce methods.

16) RINZE [105]—a system designed for data series exploration. By means of a special adaptive index data structure (ADS+) it is capable to overcome the bottleneck of the index construction by shifting the construction of the leaf nodes of the index to query time.
17) IFocus [106]—a sampling algorithm for a rapid generation of approximate visualizations preserving important properties of data. The method works so that for creating visualizations it does not use more data samples than necessary and takes much less time than conventional sampling approaches.
18) ProgressiVis [107]—a Python toolkit implementing a Progressive Analytics programming paradigm. It allows a user to check and steer the process of analysis while computations are being done. It runs modules in short batches (instead of completing algorithms one after another) producing usable results and passing them from one module to another.

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:

19) CUPID [108]—a friendly query language designed to support non-programmer integration with relational databases.
20) Traceview [109]—a system for visualization and trace files manipulation.
21) Khoros [110]—a data flow visual language allowing to create block diagrams integrating software development environment for information processing and visualization.
22) XmdvTool [111]—a software package for interactive visual exploration of multivariate data sets.
23) Table Lens [112] and FOCUS [113]—visualization systems providing table displays that present data in a relational table view, using simple graphics in the cells to communicate quantitative values.
24) Narcissus [114]—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.
25) VR-VIBE [115]—a virtual reality application intended to support the co-operative analysis of document stores.
26) XGobi [116]—a tool providing predefined visualizations intended for exploring high multidimensional data.
27) ILOG Discovery [117]—a program designed as an information visualization editor allowing browsing the visualization design space of data sets.
28) D^3 (Data Driven Documents) [37]—a representation-transparent approach to visualization for the web.
29) VisComplete [118]—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.
30) Protovis [35]—an embedded domain-specific language for constructing visualizations by composing simple graphical marks such as bars, lines and labels.
31) D^3 [37]—a system implementing a representation-transparent approach for the web visualization enabling direct inspection and manipulations. A successor of Protovis.
32) Quadrigram [119]—a data visualization web-based service launched in 2012 and based on a visual programming language.

REFERENCES

[1] J. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D. H. Rogers, and M. Petersen, "An image-based approach to extreme scale in situ visualization and analysis," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2014, pp. 424–434.
[2] B. Shneiderman, "Extreme visualization: Squeezing a billion records into a million pixels," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2008, pp. 3–12.
[3] J. Heer and S. Kandel, "Interactive analysis of big data," *ACM Crossroads*, vol. 19, no. 1, pp. 50–54, 2012. [Online]. Available: http://doi.acm.org/10.1145/2331042.2331058
[4] J. Priestley, "A chart of biography, london," *British Library, London. I*, 1765.
[5] C. J. Minard, "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*, vol. 1813, 1812.
[6] J. Snow and B. Richardson, *Snow Cholera: Being Reprint Two Papers John Snow, MD, Together Biographical Memoir BW Richardson, Introduction Wade Hampton Frost, MD*. New York, NY, USA: Hafner, 1965.
[7] E. Tufte, *Envisioning Information*. Cheshire, CT, USA: Graphics Press, 1990.
[8] J. Bertin, *Semiology of Graphics*. Madison, WI, USA: Univ. of Wisconsin Press, 1983.
[9] W. H. Benson and B. Kitous, "Interactive analysis and display of tabular data," *ACM SIGGRAPH Comput. Graph.*, vol. 11, no. 2, pp. 48–53, 1977.
[10] J. Mackinlay, "Automating the design of graphical presentations of relational Information," *ACM Trans. Graph.*, vol. 5, no. 2, pp. 110–141, 1986.
[11] S. M. Casner, "Task-analytic approach to the automated design of graphic presentations," *ACM Trans. Graph.*, vol. 10, no. 2, pp. 111–151, 1991.
[12] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *Proc. Int. Symp. Visual Languages/Human-Centric Comput. Languages Environments*, 1996, pp. 336–343.
[13] S. F. Roth and J. Mattis, "Automating the presentation of information," in *Proc. IEEE 7th Int. Conf. Artif. Intell. Appl.*, 1991, pp. 90–97.
[14] J. Goldstein, S. F. Roth, J. Kolojejchick, and J. Mattis, "A framework for knowledge-based interactive data exploration," *J. Visual Languages Comput.*, vol. 5, no. 4, pp. 339–363, 1994.
[15] D. A. Keim and H.-P. Kriegel, "Visdb: Database exploration using multidimensional visualization," *IEEE Comput. Graph. Appl.*, vol. 14, no. 5, pp. 40–49, Sept./Oct. 1994.
[16] A. Spoerri, "Infocrystal: A visual tool for information retrieval & management," in *Proc. 2nd Int. Conf. Inform. Knowl. Manag.*, 1993, pp. 11–20.
[17] C. Ahlberg and E. Wistrand, "Ivee: An information visualization and exploration environment," in *Proc. IEEE Symp. Inform. Vis.*, 1995, pp. 66–73.
[18] S. F. Roth, P. Lucas, J. A. Senn, C. C. Gomberg, M. B. Burks, P. J. Stroffolino, A. Kolojechick, and C. Dunmire, "Visage: A user interface environment for exploring information," in *Proc. IEEE Symp. Inform. Vis.*, 1996, pp. 3–12.
[19] M. Derthick, J. Kolojejchick, and S. F. Roth, "An interactive visual query environment for exploring data," in *Proc. 10th Annu. ACM Symp. User Interface Softw. Technol.*, 1997, pp. 189–198.
[20] F. Ruskey and M. Weston, "A survey of venn diagrams," *Electron. J. Combinatorics*, vol. 4, pp. 49–51, 1997.
[21] M. C. Chuah and S. F. Roth, "Visualizing common ground," in *Proc. IEEE 7th Int. Conf. Inform. Vis.*, 2003, pp. 365–372.
[22] C. Ahlberg and B. Shneiderman, "Visual information seeking: tight coupling of dynamic query filters with starfield displays," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 1994, pp. 313–317.
[23] C. Ahlberg, "Spotfire: An information exploration environment," *ACM SIGMOD Rec.*, vol. 25, no. 4, pp. 25–29, 1996.
[24] C. Ahlberg and B. Shneiderman, "The alphaslider: A compact and rapid selector," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 1994, pp. 365–371.
[25] C. Ahlberg, C. Williamson, and B. Shneiderman, "Dynamic queries for information exploration: An implementation and evaluation," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 1992, pp. 619–626.

[26] E. D. Petajan, Y. D. Jean, D. Lieuwen, and V. Anupam, "Dataspace: An automated visualization. system for large databases," in *Proc. SPIE—The Int. Soc. Opt. Eng.*, 1997, pp. 89–98.

[27] M. Stonebraker, J. Chen, N. Nathan, C. Paxson, and J. Wu, "Tioga: Providing data management support for scientific visualization applications," in *Proc. 19th Very Large Data Bases Conf.*, 1993, pp. 25–38.

[28] A. Aiken, J. Chen, M. Stonebraker, and A. Woodruff, "Tioga-2: A direct manipulation database Vis. environment," in *Proc. IEEE 29th Int. Conf. Data Eng.*, 1996, pp. 208–208.

[29] A. Woodruff, C. Olston, A. Aiken, M. Chu, V. Ercegovac, M. Lin, M. Spalding, and M. Stonebraker, "Datasplash: A direct manipulation environment for programming semantic zoom visualization of tabular data," *J. Visual Languages Comput.*, vol. 12, no. 5, pp. 551–571, 2001.

[30] C. Olsten, M. Stonebraker, A. Aiken, and J. M. Hellerstein, "Viqing: Visual interactive querying," in *Proc. IEEE Symp. Visual Languages*, 1998, pp. 162–169.

[31] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki, and K. Wenger, "Devise: Integrated querying and visual exploration of large datasets," in *ACM SIGMOD Record*, 1997, pp. 301–312.

[32] R. Bosch, C. Stolte, D. Tang, J. Gerth, M. Rosenblum, and P. Hanrahan, "Rivet: A flexible environment for computer systems Vis.," *ACM SIGGRAPH Comput. Graph.*, vol. 34, no. 1, pp. 68–73, 2000.

[33] C. Stolte, D. Tang, and P. Hanrahan, "Polaris: A system for query, analysis, and visualization of multidimensional relational databases," *IEEE Trans. Vis. Comput. Graph.*, vol. 8, no. 1, pp. 52–65, Jan./Mar. 2002.

[34] K. Morton, M. Balazinska, D. Grossman, and J. Mackinlay, "Support the data enthusiast: Challenges for next-generation data-analysis systems," *Proc. Very Large Data Bases Endowment*, vol. 7, no. 6, pp. 453–456, 2014.

[35] M. Bostock and J. Heer, "Protovis: A graphical toolkit for visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 6, pp. 1121–1128, Nov. 2009.

[36] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer, "Reactive vega: A streaming dataflow architecture for declarative interactive Vis.," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 1, pp. 659–668, Jan. 2016.

[37] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 12, pp. 2301–2309, Dec. 2011.

[38] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis, "Dremel: Interactive analysis of web-scale datasets," *Proc. Very Large Data Bases Endowment*, vol. 3, no. 1-2, pp. 330–339, 2010.

[39] T. Mostak, "An overview of mapd (massively parallel database)," in *White paper*. Massachusetts Institute of Technology, 2013.

[40] J. Varia and S. Mathew, "Overview of amazon web services," *Jan-2014*, 2013.

[41] A. Kemper and T. Neumann, "Hyper: A hybrid oltp&olap main memory database system based on virtual memory snapshots," in *Proc. IEEE 27th Int. Conf. Data Eng.*, 2011, pp. 195–206.

[42] I. S. M. Ashish Gupta, Ed., *Materialized Views*. Cambridge, MA, USA: MIT Press, 1999.

[43] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh, "Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total," in *Proc. 12th Int. Conf. Data Eng.*, 1996, pp. 152–159. [Online]. Available: http://dx.doi.org/10.1109/ICDE.1996.492099

[44] Z. Liu, B. Jiang, and J. Heer, "*imMens*: Real-time visual querying of big data," *Comput. Graph. Forum*, vol. 32, no. 3, pp. 421–430, 2013. [Online]. Available: http://dx.doi.org/10.1111/cgf.12129

[45] J. M. Hellerstein, R. Avnur, A. Chou, C. Hidber, C. Olston, V. Raman, T. Roth, and P. J. Haas, "Interactive data analysis: The control project," *IEEE Comput.*, vol. 32, no. 8, pp. 51–59, Aug. 1999. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/2.781635

[46] D. Fisher, S. M. Drucker, and A. C. König, "Exploratory Vis. involving incremental, approximate database queries and uncertainty," *IEEE Comput. Graph. Appl.*, vol. 32, no. 4, pp. 55–62, Jul./Aug. 2012. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MCG.2012.48

[47] D. Fisher, I. O. Popov, S. M. Drucker, and M. C. schraefel, "Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2012, pp. 1673–1682. [Online]. Available: http://doi.acm.org/10.1145/2207676.2208294

[48] J. Im, F. G. Villegas, and M. J. McGuffin, "Visreduce: Fast and responsive incremental information visualization of large datasets," in *Proc. IEEE Int. Conf. Big Data*, 2013, pp. 25–32. [Online]. Available: http://dx.doi.org/10.1109/BigData.2013.6691710

[49] C. Jermaine, A. Dobra, S. Arumugam, S. Joshi, and A. Pol, "The sort-merge-shrink join," *ACM Trans. Database Syst.*, vol. 31, no. 4, pp. 1382–1416, 2006. [Online]. Available: http://doi.acm.org/10.1145/1189775

[50] X. Li, J. Han, Z. Yin, J. Lee, and Y. Sun, "Sampling cube: A framework for statistical OLAP over sampling data," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2008, pp. 779–790. [Online]. Available: http://doi.acm.org/10.1145/1376616.1376695

[51] S. Chaudhuri, G. Das, and V. R. Narasayya, "Optimized stratified sampling for approximate query processing," *ACM Trans. Database Syst.*, vol. 32, no. 2, p. 9, 2007. [Online]. Available: http://doi.acm.org/10.1145/1242524.1242526

[52] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy, "Join synopses for approximate query answering," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 1999, pp. 275–286. [Online]. Available: http://doi.acm.org/10.1145/304182.304207

[53] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica, "Blinkdb: Queries with bounded errors and bounded response times on very large data," in *Proc. 8th Eurosys Conf.*, 2013, pp. 29–42. [Online]. Available: http://doi.acm.org/10.1145/2465351.2465355

[54] P. R. Doshi, E. A. Rundensteiner, and M. O. Ward, "Prefetching for visual data exploratio," in *Proc. 8th Int. Conf. Database Syst. Adv. Appl.*, 2003, pp. 195–202. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/DASFAA.2003.1192383

[55] S. Chan, L. Xiao, J. Gerth, and P. Hanrahan, "Maintaining interactivity while exploring massive time series," in *Proc. IEEE Symp. Visual Analytics Sci. Technol.*, 2008, pp. 59–66. [Online]. Available: http://dx.doi.org/10.1109/VAST.2008.4677357

[56] S. Chaudhuri, R. Motwani, and V. R. Narasayya, "On random sampling over joins," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 1999, pp. 263–274.

[57] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi, "Distributed and interactive cube exploration," in *Proc. IEEE 30th Int. Conf. Data Eng.*, 2014, pp. 472–483. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2014.6816674

[58] D. Fisher, "Hotmap: Looking at geographic attention," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 6, pp. 1184–1191, Nov./Dec. 2007. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TVCG.2007.70561

[59] P. J. Haas and J. M. Hellerstein, "Ripple joins for online aggregation," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 1999, pp. 287–298. [Online]. Available: http://doi.acm.org/10.1145/304182.304208

[60] S. Acharya, P. B. Gibbons, and V. Poosala, "Congressional samples for approximate answering of group-by queries," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2000, pp. 487–498. [Online]. Available: http://doi.acm.org/10.1145/342009.335450

[61] J. Gryz, J. Guo, L. Liu, and C. Zuzarte, "Query sampling in DB2 universal database," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2004, pp. 839–843. [Online]. Available: http://doi.acm.org/10.1145/1007568.1007664

[62] M. Livny, R. Ramakrishnan, K. S. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki, and R. K. Wenger, "Devise: Integrated querying and visualization of large datasets," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 1997, pp. 301–312. [Online]. Available: http://doi.acm.org/10.1145/253260.253335

[63] U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl, "Faster visual analytics through pixel-perfect aggregation," *Proc. Very Large Data Bases Endowment*, vol. 7, no. 13, pp. 1705–1708, 2014.

[64] L. Battle, M. Stonebraker, and R. Chang, "Dynamic reduction of query result sets for interactive visualizaton," in *Proc. IEEE Int. Conf. Big Data*, 2013, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/BigData.2013.6691708

[65] E. Wu, L. Battle, and S. R. Madden, "The case for data visualization management systems [vision paper]," *Proc. Very Large Data Bases Endowment*, vol. 7, no. 10, pp. 903–906, 2014.

[66] E. Wu and A. Nandi, "Towards perception-aware interactive data visualization systems," in *Data Syst. Interactive Anal. Workshop 2015*, Chicago, IL, 2015.

[67] A. Dziedzic, J. Duggan, A. J. Elmore, V. Gadepally, and M. Stonebraker, "Bigdawg: a polystore for diverse interactive applications," in *Data Syst. Interactive Anal. Workshop 2015*, Chicago, IL, 2015.

[68] R. Wesley, M. Eldridge, and P. T. Terlecki, "An analytic data engine for Vis. in tableau," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2011, pp. 1185–1194.

[69] R. M. G. Wesley and P. Terlecki, "Leveraging compression in the tableau data engine," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2014, pp. 563–573.

[70] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik, "C-store: A column-oriented DBMS," in *Proc. 31st Int. Conf. Very Large Data Bases*,, 2005, pp. 553–564. [Online]. Available: http://www.vldb2005.org/program/paper/thu/p553-stonebraker.pdf

[71] S. I. F. G. N. Nes and S. M. S. M. M. Kersten, "Monetdb: Two decades of research in column-oriented database architectures," *IEEE Data Eng. Bull.*, vol. 35, no. 1, p. 40, 2012.

[72] D. Fisher, "Incremental, approximate database queries and uncertainty for exploratory visualization," in *Proc. IEEE Symp. Large Data Anal. Vis.*, 2011, pp. 73–80. [Online]. Available: http://dx.doi.org/10.1109/LDAV.2011.6092320

[73] N. Elmqvist and J. Fekete, "Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines," *IEEE Trans. Vis. Comput. Graph.*, vol. 16, no. 3, pp. 439–454, 2010.

[74] L. Lins, J. T. Klosowski, and C. Scheidegger, "Nanocubes for real-time exploration of spatiotemporal datasets," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 12, pp. 2456–2465, Dec. 2013.

[75] P. Godfrey, J. Gryz, P. Lasek, and Razvi, "Visualization through inductive aggregation," in *Proc. Int. Conf. Extending Database Technol.*, 2016, pp. 600–603.

[76] A. Streit, B. Pham, and R. Brown, "A spreadsheet approach to facilitate visiualization of uncertainty in information," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 1, pp. 61–72, Jan./Feb. 2008. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TVCG.2007.70426

[77] C. Olston and J. D. Mackinlay, "Visualizing data with bounded uncertainty," in *Proc. IEEE Symp. Inform. Vis.*, 2002, pp. 37–40. [Online]. Available: http://dx.doi.org/10.1109/INFVIS.2002.1173145

[78] R. Kosara, S. Miksch, and H. Hauser, "Semantic depth of field," in *Proc. IEEE Symp. Inform. Vis.*, 2001, pp. 97–104. [Online]. Available: http://dx.doi.org/10.1109/INFVIS.2001.963286

[79] C. M. Wittenbrink, A. Pang, and S. K. Lodha, "Glyphs for visualizing uncertainty in vector fields," *IEEE Trans. Vis. Comput. Graph.*, vol. 2, no. 3, pp. 266–279, Sep. 1996. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/2945.537309

[80] J. Sanyal, S. Zhang, G. Bhattacharya, P. Amburn, and R. J. Moorhead, "A user study to compare four uncertainty visualization methods for 1D and 2D datasets," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 6, pp. 1209–1218, Nov./Dec. 2009. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TVCG.2009.114

[81] Y. Wu, B. Harb, J. Yang, and C. Yu, "Efficient evaluation of object-centric exploration queries for visualization," *Proc. 41st Int. Conf. Very Large Data Bases*, 2015, pp. 1752–1763.

[82] E. Bertini and G. Santucci, "Give chance a chance- modeling density to enhance scatter plot quality through random data sampling," *Inform. Vis.*, vol. 5, no. 2, pp. 95–110, 2006. [Online]. Available: http://dx.doi.org/10.1057/palgrave.ivs.9500122

[83] M. Ward, W. Peng, and X. Wang, "Hierarchical visual data mining for large-scale data," *Comput. Statist.*, vol. 9, no. 1, pp. 147–158, 2004.

[84] H. Wickham, "Bin-summarise-smooth: A framework for visualising large data," had.co.nz, 2013.

[85] R. A. Rensink and G. Baldridge, "The perception of correlation in scatterplots," *Comput. Graph. Forum*, vol. 29, no. 3. pp. 1203–1210.

[86] L. Harrison, F. Yang, S. Franconeri, and R. Chang, "Ranking visualization of correlation using weber's law," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 12, pp. 1943–1952, Dec. 2014.

[87] J. Heer and B. Shneiderman, "Interactive dynamics for visual analysis," *Commun. ACM*, vol. 55, no. 4, pp. 45–54, 2012. [Online]. Available: http://doi.acm.org/10.1145/2133806.2133821

[88] K. Sato, "An inside look at google bigquery," White paper, URL: https://cloud. google. com/files/BigQueryTechnicalWP. pdf, 2012.

[89] M. M. Zloof, "Query by example," in *Proc. May 19-22, 1975, Nat. Comput. Conf. Expo.*, 1975, pp. 431–438.

[90] C. F. Herot, "Spatial management of data," *ACM Trans. Database Syst.*, vol. 5, no. 4, pp. 493–513, 1980.

[91] R. Cattell, "An entity-based database user interface," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 1980, pp. 144–150.

[92] H. K. Wong, I. Kuo, et al., "Guide: Graphical user interface for database exploration." in *Proc. Int. Conf. Very Large Data Bases*, 1982, pp. 22–32.

[93] C. Upson, T. A. Faulhaber Jr, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. Van Dam, "The application visualization system: A computational environment for scientific visualization," *IEEE Comput. Graph. Appl.*, vol. 9, no. 4, pp. 30–42, Jul. 1989.

[94] D. A. Bailey, J. E. Cuny, and C. P. Loomis, "Paragraph: Graph editor support for parallel programming environments," *Int. J. Parallel Program.*, vol. 19, no. 2, pp. 75–110, 1990.

[95] W. Hibbard, J. Kellum, and B. Paul, "Vis5d version 5.2," *Vis. Project, University of Wisconsin–Madison Space Sci. Eng. Center*, 1990.

[96] M. Hemmje, C. Kunkel, and A. Willett, "Lyberworlda visualization user interface supporting fulltext retrieval," in *Proc. 17th Annu. Int. Conf. Res. Develop. Inform. Retrieval*, 1994, pp. 249–259.

[97] P. C. Wong, B. Hetzler, C. Posse, M. Whiting, S. Havre, N. Cramer, A. Shah, M. Singhal, A. Turner, and J. Thomas, "In-spire infovis 2004 contest entry," in *Proc. IEEE Symp. Inform. Vis.*, 2004, pp. r2–r2.

[98] A. S. Jacobson, A. L. Berkin, and M. N. Orton, "Linkwinds: Interactive scientific data analysis and visualization," *Commun. ACM*, vol. 37, no. 4, pp. 42–52, 1994.

[99] T. C. Sprenger, M. H. Gross, D. Bielser, and T. Strasser, "Ivory—An object-oriented framework for physics-based information Vis. in java," in *Proc. IEEE Symp. Inform. Vis.*, 1998, pp. 79–86.

[100] G. Dennis Jr, B. T. Sherman, D. A. Hosack, J. Yang, W. Gao, H. C. Lane, R. A. Lempicki, et al., "David: Database for annotation, visualization, and integrated discovery," *Genome Biol.*, vol. 4, no. 5, p. P3, 2003.

[101] W. Bethel, C. Siegerist, J. Shalf, P. Shetty, T. Jankun-Kelly, O. Kreylos, and K.-L. Ma, "Visportal: Deploying grid-enabled visualization tools through a web-portal interface," Lawrence Berkeley National Laboratory, Berkeley, CA, USA, Tech. Rep. LBNL-52940, 2003.

[102] S. G. Eick, M. A. Eick, J. Fugitt, B. Horst, M. Khailo, and R. A. Lankenau, "Thin client visualization," in *Proc. IEEE Symp. Visual Analytics Sci. Technol.*, 2007, pp. 51–58.

[103] S. Kandel, R. Parikh, A. Paepcke, J. M. Hellerstein, and J. Heer, "Profiler: Integrated statistical analysis and visualization for data quality assessment," in *Proc. Int. Working Conf. Adv. Visual Interfaces*, 2012, pp. 547–554.

[104] A. Eldawy, M. F. Mokbel, and C. Jonathan, "A demonstration of hadoopviz: An extensible mapreduce system for visualizing big spatial data," in *Proc. 41st Int. Conf. Very large Data Bases Endowment*, 2015, pp. 1896–1899.

[105] K. Zoumpatianos, S. Idreos, and T. Palpanas, "Rinse: Interactive data series exploration with ADS," *Proc. Very large Data Bases Endowment*, vol. 8, no. 12, pp. 1912–1915, 2015.

[106] A. Kim, E. Blais, A. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld, "Rapid sampling for visualization with ordering guarantees," *Proc. Very large Data Bases Endowment*, vol. 8, no. 5, pp. 521–532, 2015.

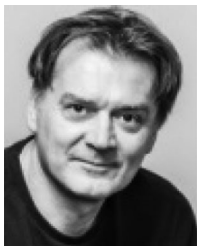
[107] J.-D. Fekete, "Progressivis: A toolkit for steerable progressive analytics and visualization," in *Proc. 1st Workshop Data Syst. Interactive Anal.*, 2015, p. 5.

[108] N. H. McDonald and M. Stonebraker, "Cupid-the friendly query language." in *Proc. Int. Conf. ACM Pacific*, 1975, pp. 127–131.

[109] A. D. Malony, D. H. Hammerslag, and D. J. Jablonowski, *Traceview: Trace visualization tool*," *Proc. 1st Int. ACPC Conf. Parallel Comput.*, 1992, pp. 102–114.

[110] K. Konstantinides and J. R. Rasure, "The khoros software development environment for image and signal processing," *IEEE Trans. Image Process.*, vol. 3, no. 3, pp. 243–252, May 1994.
[111] M. O. Ward, "Xmdvtool: Integrating multiple methods for visualizing multivariate data," in *Proc. Conf. Vis.*, 1994, pp. 326–333.
[112] R. Rao and S. K. Card, "The table lens: Merging graphical and symbolic representations in an interactive focus+ context visualization for tabular information," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 1994, pp. 318–322.
[113] M. Spenke, C. Beilken, and T. Berlage, "Focus: The interactive table for product comparison and selection," in *Proc. 9th Annu. ACM Symp. User Interface Softw. Technol.*, 1996, pp. 41–50.
[114] R. J. Hendley, N. S. Drew, A. M. Wood, and R. Beale, "Case study narcissus: Visualising information," in *Proc. IEEE Symp. Inform. Vis.*, 1995, pp. 90–96.
[115] S. Benford, D. Snowdon, C. Greenhalgh, R. Ingram, I. Knox, and C. Brown, "Vr-vibe: A virtual environment for co-operative information retrieval," *Comput. Graph. Forum*, vol. 14, no. 3, pp. 349–360, 1995.
[116] A. Buja, D. Cook, and D. F. Swayne, "Interactive high-dimensional data visualization," *J. Comput. Graphical Statist.*, vol. 5, no. 1, pp. 78–99, 1996.
[117] T. Baudel, "Browsing through an information visualization design space," in *Proc. Int. Conf. Extended Abstracts Human Factors Comput. Syst.*, 2004, pp. 765–766.
[118] D. Koop, C. E. Scheidegger, S. P. Callahan, J. Freire, and C. T. Silva, "Viscomplete: Automating suggestions for visualization pipelines," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 6, pp. 1691–1698, Nov./Dec. 2008.
[119] A. Satyanarayan, K. Wongsuphasawat, and J. Heer, "Declarative interaction design for data visualization," in *Proc. 27th Annu. ACM Symp. User Interface Softw. Technol.*, 2014, pp. 669–678.
[120] X. Hu, T. Y. Lin, V. Raghavan, B. W. Wah, R. A. Baeza-Yates, G. Fox, C. Shahabi, M. Smith, Q. Yang, R. Ghani, W. Fan, R. Lempel, and R. Nambiar, Eds., *Proc. 2013 IEEE Int. Conf. Big Data*, 6-9 Oct. 2013, Santa Clara, CA, USA, IEEE, 2013.
[121] A. Delis, C. Faloutsos, and S. Ghandeharizadeh, Eds., *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 1999.

Parke Godfrey received the PhD degree in computer science from the University of Maryland, USA, in 1999. He is currently an associate professor in the Department of Computer Science and Engineering at York University, Canada. His research interests include databases, with specific interests in cooperative query answering, reasoning over integrity constraints, logic-based query optimization, semantic caching, and knowledge discovery in databases.



Jarek Gryz received the PhD degree in computer science from the University of Maryland, USA, in 1997. He is currently a professor in the Department of Computer Science Engineering at the York University, Canada. His main research interests include database systems, data mining, query optimization via data mining, preference queries, and query sampling.

Piotr Lasek received the MSc degree in computer science and the PhD degree from the Warsaw University of Technology in 2004 and 2012, respectively. He is curently a postdoctoral fellow at York University, Canada, and an assistant professor at Rzeszów University, Poland. His main areas of research include data mining and interactive data visualization.