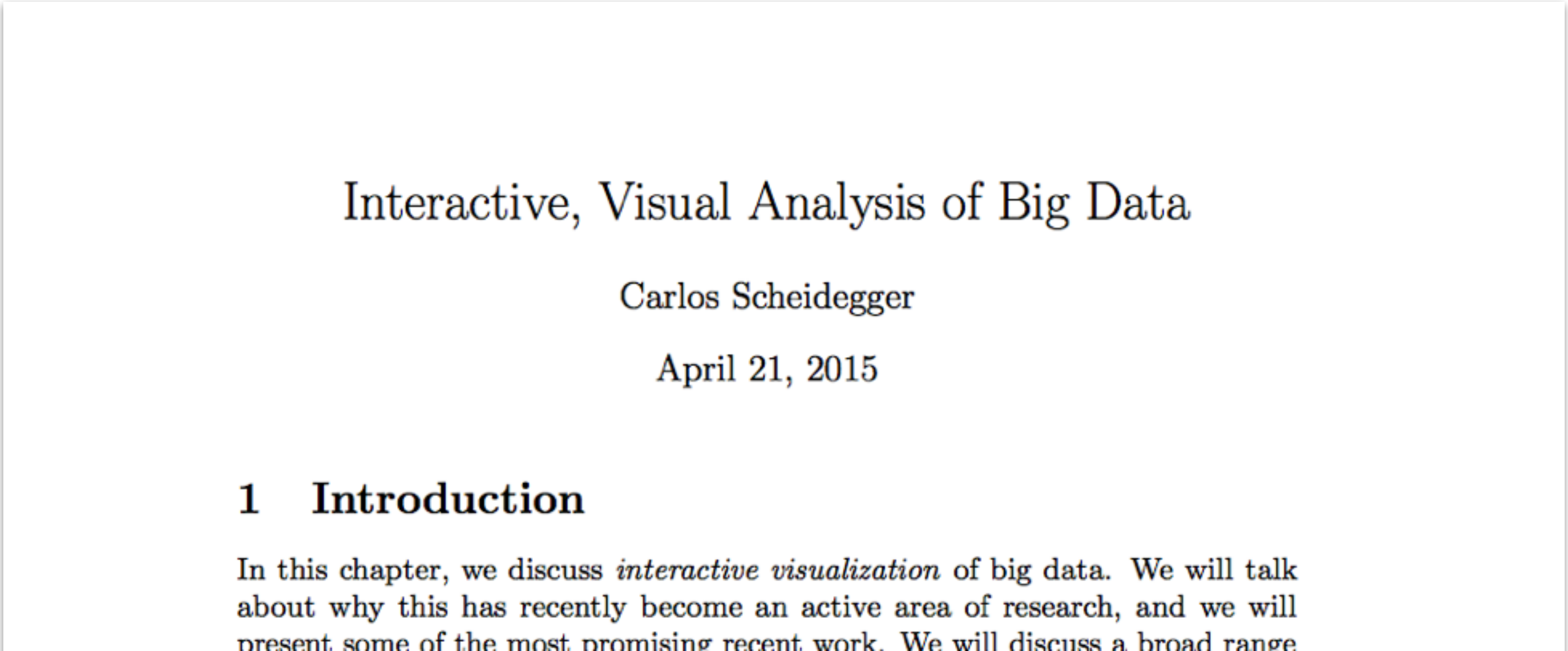
Data Cubes, Aggregation Operations

CSC 630

Reading for next few weeks: **all** papers mentioned in this lecture

http://cscheid.net/static/unlinked/big_data_visualization.pdf



Interactive, Visual Analysis of Big Data

Carlos Scheidegger

April 21, 2015

1 Introduction

In this chapter, we discuss *interactive visualization* of big data. We will talk about why this has recently become an active area of research, and we will present some of the most promising recent work. We will discuss a broad range

Next week: Data Cubes, and **you will present**

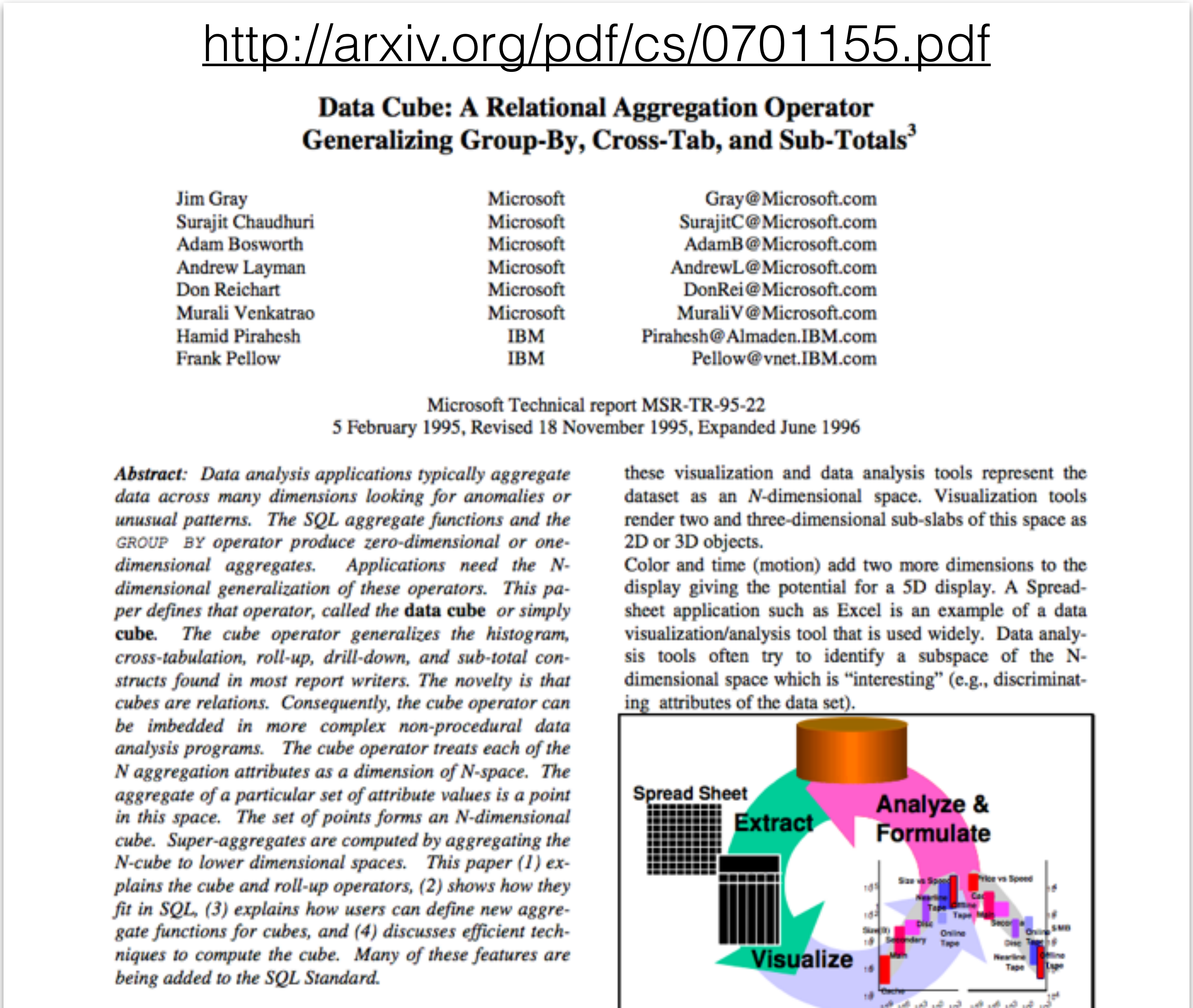
Following week: progressive visualizations

Why do we care about aggregation operations?

(ggplot demos)

SALES			
Model	Year	Color	Sales
Chevy	1990	red	5
Chevy	1990	white	87
Chevy	1990	blue	62
Chevy	1991	red	54
Chevy	1991	white	95
Chevy	1991	blue	49
Chevy	1992	red	31
Chevy	1992	white	54
Chevy	1992	blue	71
Ford	1990	red	64
Ford	1990	white	62
Ford	1990	blue	63
Ford	1991	red	52
Ford	1991	white	9
Ford	1991	blue	55
Ford	1992	red	27
Ford	1992	white	62
Ford	1992	blue	39

Data Cubes



http://arxiv.org/pdf/cs/0701155.pdf

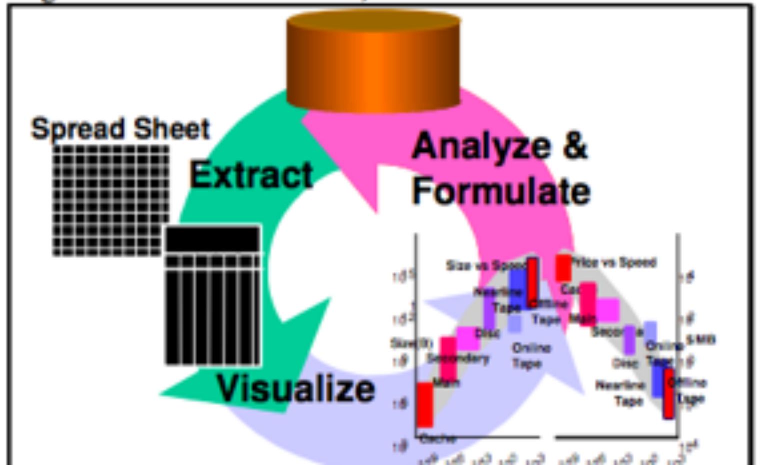
Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals[3]

Jim Gray	Microsoft	Gray@Microsoft.com
Surajit Chaudhuri	Microsoft	SurajitC@Microsoft.com
Adam Bosworth	Microsoft	AdamB@Microsoft.com
Andrew Layman	Microsoft	AndrewL@Microsoft.com
Don Reichart	Microsoft	DonRei@Microsoft.com
Murali Venkatrao	Microsoft	MuraliV@Microsoft.com
Hamid Pirahesh	IBM	Pirahesh@Almaden.IBM.com
Frank Pellow	IBM	Pellow@vnet.IBM.com

Microsoft Technical report MSR-TR-95-22
5 February 1995, Revised 18 November 1995, Expanded June 1996

Abstract: *Data analysis applications typically aggregate data across many dimensions looking for anomalies or unusual patterns. The SQL aggregate functions and the* `GROUP BY` *operator produce zero-dimensional or one-dimensional aggregates. Applications need the N-dimensional generalization of these operators. This paper defines that operator, called the* **data cube** *or simply* **cube**. *The cube operator generalizes the histogram, cross-tabulation, roll-up, drill-down, and sub-total constructs found in most report writers. The novelty is that cubes are relations. Consequently, the cube operator can be imbedded in more complex non-procedural data analysis programs. The cube operator treats each of the N aggregation attributes as a dimension of N-space. The aggregate of a particular set of attribute values is a point in this space. The set of points forms an N-dimensional cube. Super-aggregates are computed by aggregating the N-cube to lower dimensional spaces. This paper (1) explains the cube and roll-up operators, (2) shows how they fit in SQL, (3) explains how users can define new aggregate functions for cubes, and (4) discusses efficient techniques to compute the cube. Many of these features are being added to the SQL Standard.*

these visualization and data analysis tools represent the dataset as an *N*-dimensional space. Visualization tools render two and three-dimensional sub-slabs of this space as 2D or 3D objects.
Color and time (motion) add two more dimensions to the display giving the potential for a 5D display. A Spreadsheet application such as Excel is an example of a data visualization/analysis tool that is used widely. Data analysis tools often try to identify a subspace of the N-dimensional space which is "interesting" (e.g., discriminating attributes of the data set).

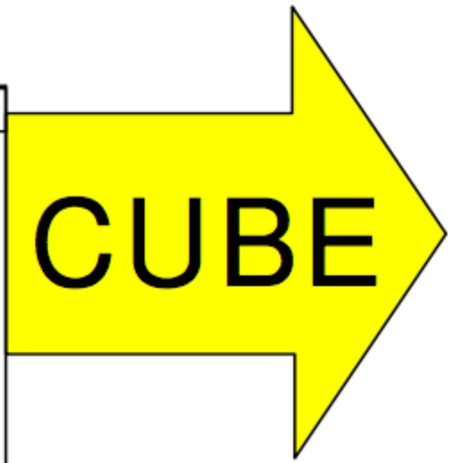


SALES			
Model	Year	Color	Sales
Chevy	1990	red	5
Chevy	1990	white	87
Chevy	1990	blue	62
Chevy	1991	red	54
Chevy	1991	white	95
Chevy	1991	blue	49
Chevy	1992	red	31
Chevy	1992	white	54
Chevy	1992	blue	71
Ford	1990	red	64
Ford	1990	white	62
Ford	1990	blue	63
Ford	1991	red	52
Ford	1991	white	9
Ford	1991	blue	55
Ford	1992	red	27
Ford	1992	white	62
Ford	1992	blue	39

```
SELECT Model, Year, Color, SUM(sales) AS Sales
FROM Sales
WHERE Model in {'Ford', 'Chevy'}
  AND Year BETWEEN 1990 AND 1992
GROUP BY CUBE Model, Year, Color;
```

SALES

Model	Year	Color	Sales
Chevy	1990	red	5
Chevy	1990	white	87
Chevy	1990	blue	62
Chevy	1991	red	54
Chevy	1991	white	95
Chevy	1991	blue	49
Chevy	1992	red	31
Chevy	1992	white	54
Chevy	1992	blue	71
Ford	1990	red	64
Ford	1990	white	62
Ford	1990	blue	63
Ford	1991	red	52
Ford	1991	white	9
Ford	1991	blue	55
Ford	1992	red	27
Ford	1992	white	62
Ford	1992	blue	39



DATA CUBE

Model	Year	Color	Sales
Chevy	1990	blue	62
Chevy	1990	red	5
Chevy	1990	white	95
Chevy	1990	ALL	154
Chevy	1991	blue	49
Chevy	1991	red	54
Chevy	1991	white	95
Chevy	1991	ALL	198
Chevy	1992	blue	71
Chevy	1992	red	31
Chevy	1992	white	54
Chevy	1992	ALL	156
Chevy	ALL	blue	182
Chevy	ALL	red	90
Chevy	ALL	white	236
Chevy	ALL	ALL	508
Ford	1990	blue	63
Ford	1990	red	64
Ford	1990	white	62
Ford	1990	ALL	189
Ford	1991	blue	55
Ford	1991	red	52
Ford	1991	white	9
Ford	1991	ALL	116
Ford	1992	blue	39
Ford	1992	red	27
Ford	1992	white	62
Ford	1992	ALL	128
Ford	ALL	blue	157
Ford	ALL	red	143
Ford	ALL	white	133
Ford	ALL	ALL	433
ALL	1990	blue	125
ALL	1990	red	69
ALL	1990	white	149
ALL	1990	ALL	343
ALL	1991	blue	106
ALL	1991	red	104
ALL	1991	white	110
ALL	1991	ALL	314
ALL	1992	blue	110
ALL	1992	red	58
ALL	1992	white	116
ALL	1992	ALL	284
ALL	ALL	blue	339
ALL	ALL	red	233
ALL	ALL	white	369
ALL	ALL	ALL	941

Figure 4: A 3D data cube (right) built from the table at the left by the `CUBE` statement at the top of the figure.

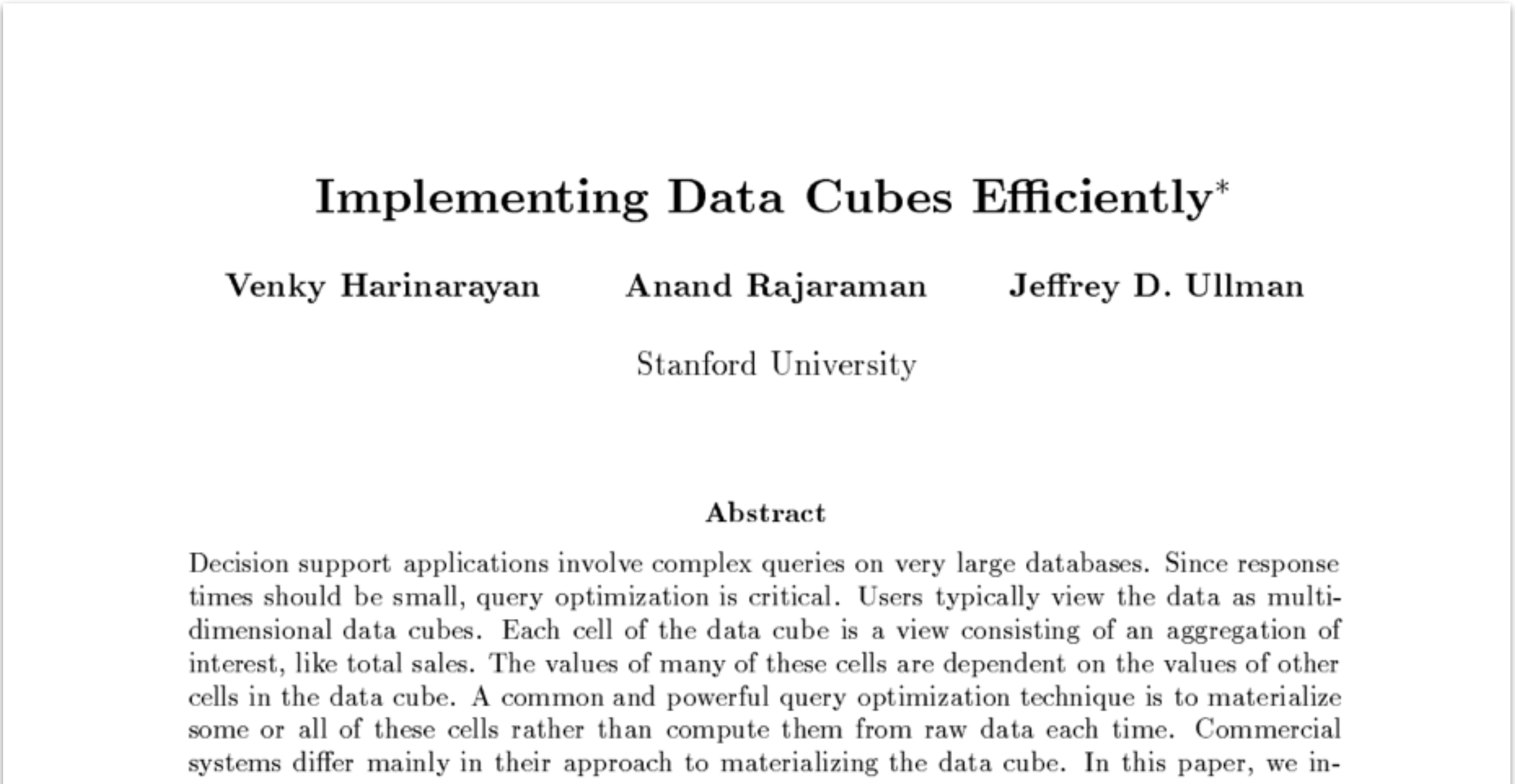
How do we compute it?

How do we compute it?

- Either on the fly
- Or accessing precomputed values
- Or some combination of it

How do we compute it?

http://web.eecs.umich.edu/~jag/eecs584/papers/implementing_data_cube.pdf



Implementing Data Cubes Efficiently*

Venky Harinarayan   Anand Rajaraman   Jeffrey D. Ullman

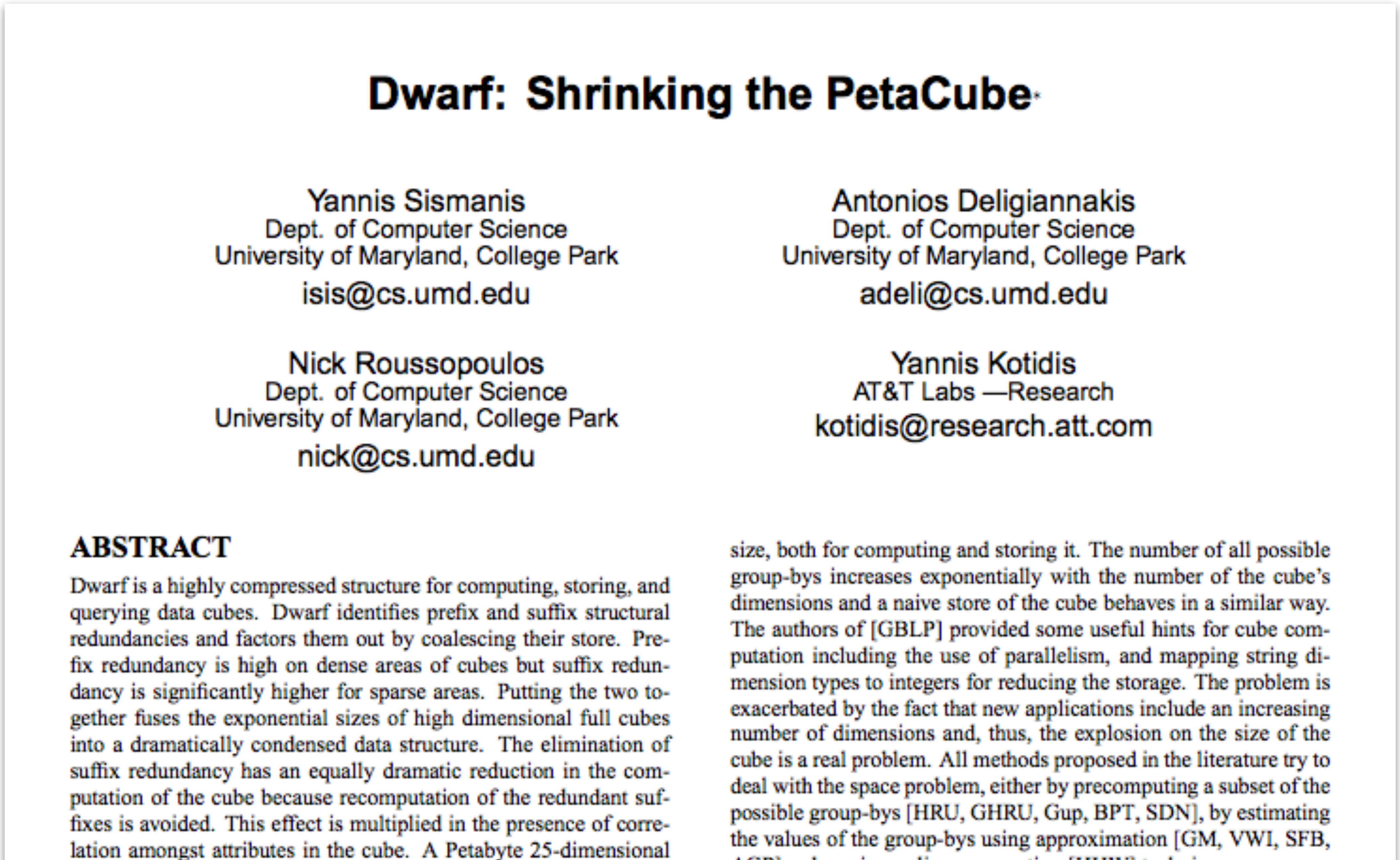
Stanford University

Abstract

Decision support applications involve complex queries on very large databases. Since response times should be small, query optimization is critical. Users typically view the data as multi-dimensional data cubes. Each cell of the data cube is a view consisting of an aggregation of interest, like total sales. The values of many of these cells are dependent on the values of other cells in the data cube. A common and powerful query optimization technique is to materialize some or all of these cells rather than compute them from raw data each time. Commercial systems differ mainly in their approach to materializing the data cube. In this paper, we in-

How do we compute it?

http://www.cs.umd.edu/~nick/projects/Dwarf.pdf



Dwarf: Shrinking the PetaCube*

Yannis Sismanis
Dept. of Computer Science
University of Maryland, College Park
isis@cs.umd.edu

Antonios Deligiannakis
Dept. of Computer Science
University of Maryland, College Park
adeli@cs.umd.edu

Nick Roussopoulos
Dept. of Computer Science
University of Maryland, College Park
nick@cs.umd.edu

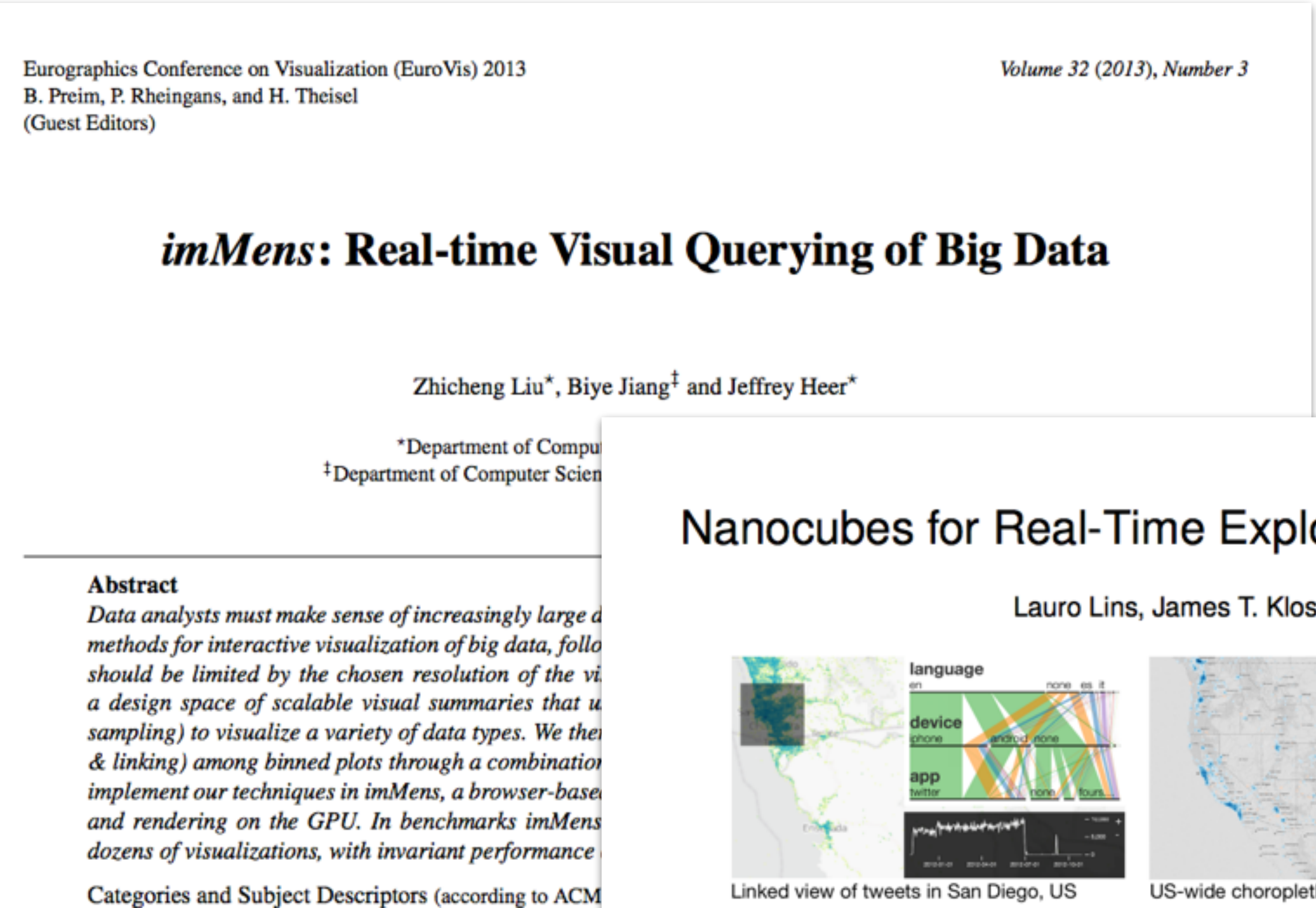
Yannis Kotidis
AT&T Labs —Research
kotidis@research.att.com

ABSTRACT

Dwarf is a highly compressed structure for computing, storing, and querying data cubes. Dwarf identifies prefix and suffix structural redundancies and factors them out by coalescing their store. Prefix redundancy is high on dense areas of cubes but suffix redundancy is significantly higher for sparse areas. Putting the two together fuses the exponential sizes of high dimensional full cubes into a dramatically condensed data structure. The elimination of suffix redundancy has an equally dramatic reduction in the computation of the cube because recomputation of the redundant suffixes is avoided. This effect is multiplied in the presence of correlation amongst attributes in the cube. A Petabyte 25-dimensional

size, both for computing and storing it. The number of all possible group-bys increases exponentially with the number of the cube's dimensions and a naive store of the cube behaves in a similar way. The authors of [GBLP] provided some useful hints for cube computation including the use of parallelism, and mapping string dimension types to integers for reducing the storage. The problem is exacerbated by the fact that new applications include an increasing number of dimensions and, thus, the explosion on the size of the cube is a real problem. All methods proposed in the literature try to deal with the space problem, either by precomputing a subset of the possible group-bys [HRU, GHRU, Gup, BPT, SDN], by estimating the values of the group-bys using approximation [GM, VWI, SFB,

How do we compute it?



Eurographics Conference on Visualization (EuroVis) 2013
B. Preim, P. Rheingans, and H. Theisel
(Guest Editors)

Volume 32 (2013), Number 3

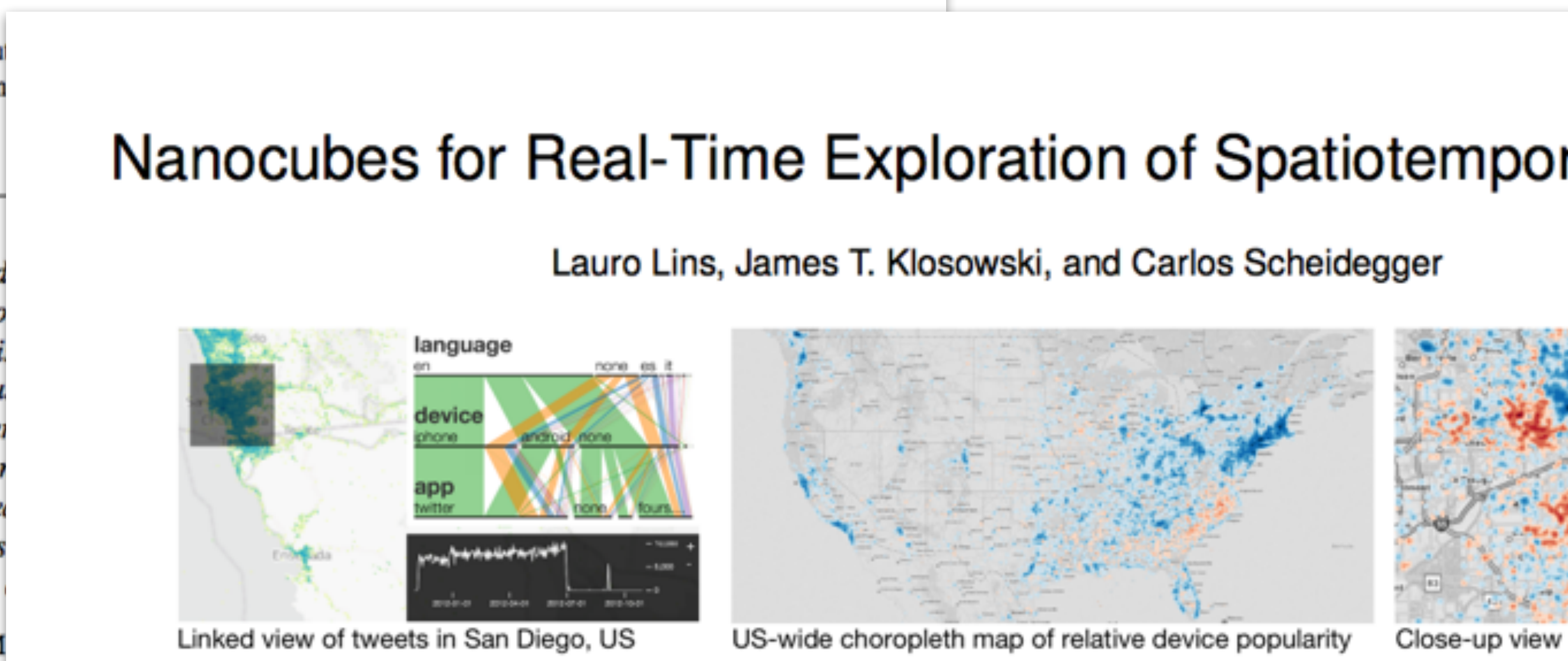
imMens: Real-time Visual Querying of Big Data

Zhicheng Liu[*], Biye Jiang[‡] and Jeffrey Heer[*]

[*]Department of Compu
[‡]Department of Computer Scien

Abstract
Data analysts must make sense of increasingly large d
methods for interactive visualization of big data, follo
should be limited by the chosen resolution of the vi
a design space of scalable visual summaries that u
sampling) to visualize a variety of data types. We the
& linking) among binned plots through a combinatio
implement our techniques in imMens, a browser-base
and rendering on the GPU. In benchmarks imMens
dozens of visualizations, with invariant performance

Categories and Subject Descriptors (according to ACM



Nanocubes for Real-Time Exploration of Spatiotempo

Lauro Lins, James T. Klosowski, and Carlos Scheidegger

Linked view of tweets in San Diego, US
US-wide choropleth map of relative device popularity
Close-up view

Different kinds of aggregation operations

- What’s the main difference between “a+b” and max(a,b)?

Different kinds of aggregation operations

- What’s the main difference between “a+b” and max(a,b)?
  - **Groups** vs **monoids**

Spatial Tricks

- Question: Given a 1D array of n integers and O(n) precomputation time and space, how do you answer queries about the sum of k consecutive integers in O(1) per query?
- Question: Given a 1D array of n integers and O(n) precomputation time and space, how do you answer queries about the max of k consecutive integers in O(log n) per query?

Spatial Tricks

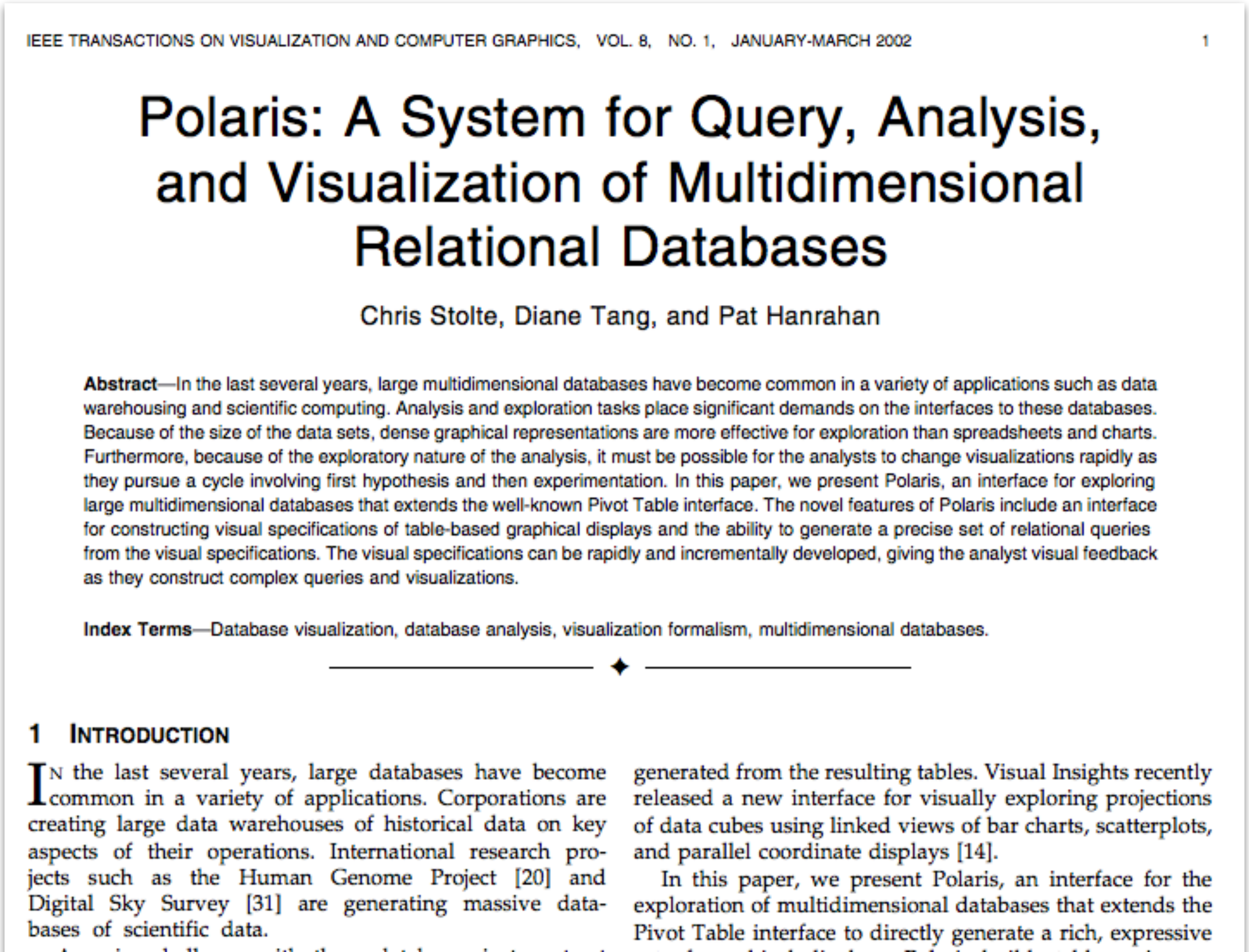
- How do you extend these tricks to multiple dimensions?

Spatial Tricks

- How do you extend these tricks to multiple dimensions?

How do we display it?

http://graphics.stanford.edu/papers/polaris_extended/polaris.pdf



IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, VOL. 8, NO. 1, JANUARY-MARCH 2002 1

Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases

Chris Stolte, Diane Tang, and Pat Hanrahan

Abstract—In the last several years, large multidimensional databases have become common in a variety of applications such as data warehousing and scientific computing. Analysis and exploration tasks place significant demands on the interfaces to these databases. Because of the size of the data sets, dense graphical representations are more effective for exploration than spreadsheets and charts. Furthermore, because of the exploratory nature of the analysis, it must be possible for the analysts to change visualizations rapidly as they pursue a cycle involving first hypothesis and then experimentation. In this paper, we present Polaris, an interface for exploring large multidimensional databases that extends the well-known Pivot Table interface. The novel features of Polaris include an interface for constructing visual specifications of table-based graphical displays and the ability to generate a precise set of relational queries from the visual specifications. The visual specifications can be rapidly and incrementally developed, giving the analyst visual feedback as they construct complex queries and visualizations.

Index Terms—Database visualization, database analysis, visualization formalism, multidimensional databases.

1 Introduction

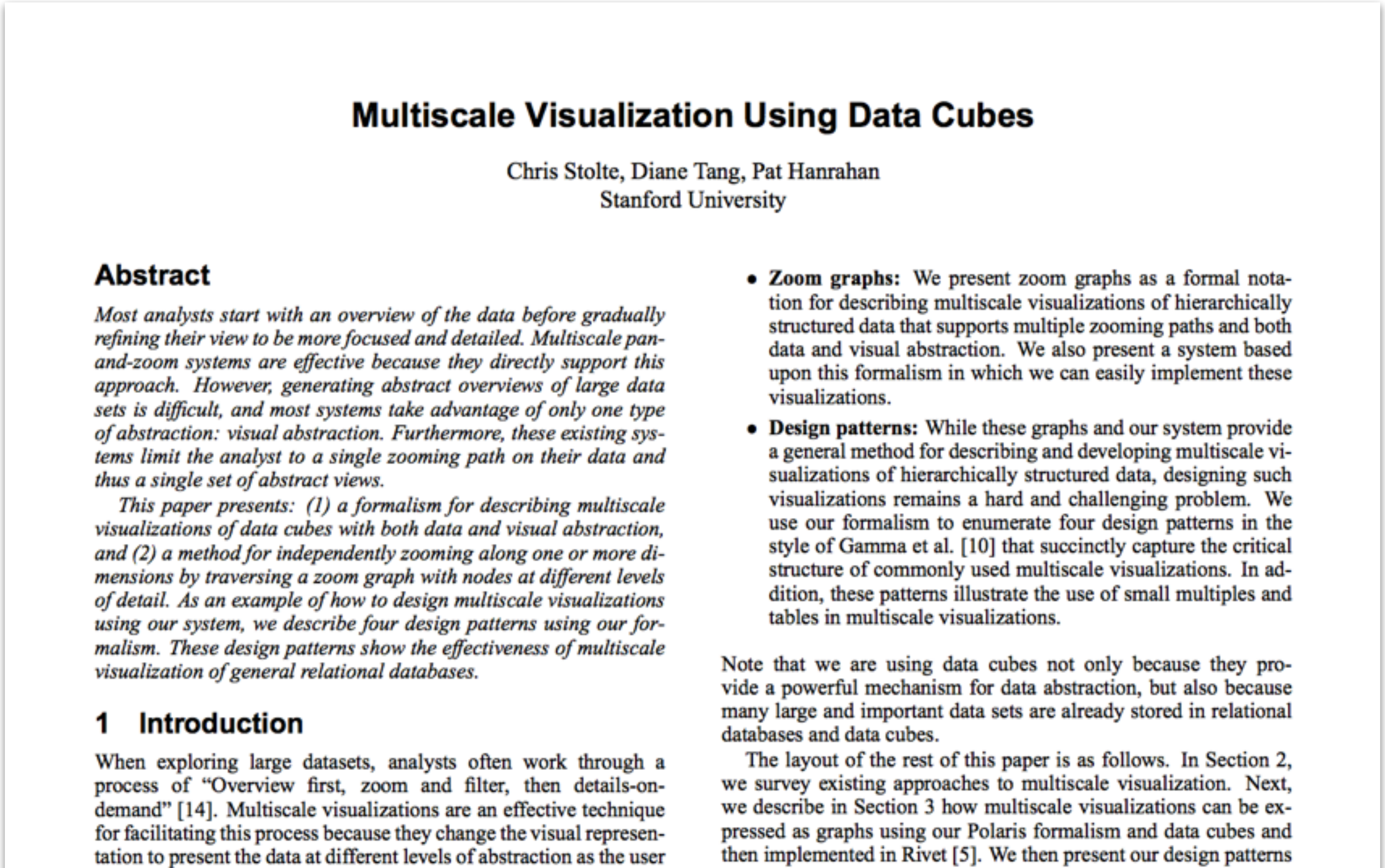
In the last several years, large databases have become common in a variety of applications. Corporations are creating large data warehouses of historical data on key aspects of their operations. International research projects such as the Human Genome Project [20] and Digital Sky Survey [31] are generating massive databases of scientific data.

generated from the resulting tables. Visual Insights recently released a new interface for visually exploring projections of data cubes using linked views of bar charts, scatterplots, and parallel coordinate displays [14].

In this paper, we present Polaris, an interface for the exploration of multidimensional databases that extends the Pivot Table interface to directly generate a rich, expressive

How do we display it?

http://graphics.stanford.edu/papers/pan_zoom/paper.pdf



Multiscale Visualization Using Data Cubes

Chris Stolte, Diane Tang, Pat Hanrahan
Stanford University

Abstract

Most analysts start with an overview of the data before gradually refining their view to be more focused and detailed. Multiscale pan-and-zoom systems are effective because they directly support this approach. However, generating abstract overviews of large data sets is difficult, and most systems take advantage of only one type of abstraction: visual abstraction. Furthermore, these existing systems limit the analyst to a single zooming path on their data and thus a single set of abstract views.

This paper presents: (1) a formalism for describing multiscale visualizations of data cubes with both data and visual abstraction, and (2) a method for independently zooming along one or more dimensions by traversing a zoom graph with nodes at different levels of detail. As an example of how to design multiscale visualizations using our system, we describe four design patterns using our formalism. These design patterns show the effectiveness of multiscale visualization of general relational databases.

1 Introduction

When exploring large datasets, analysts often work through a process of "Overview first, zoom and filter, then details-on-demand" [14]. Multiscale visualizations are an effective technique for facilitating this process because they change the visual representation to present the data at different levels of abstraction as the user

- **Zoom graphs:** We present zoom graphs as a formal notation for describing multiscale visualizations of hierarchically structured data that supports multiple zooming paths and both data and visual abstraction. We also present a system based upon this formalism in which we can easily implement these visualizations.
- **Design patterns:** While these graphs and our system provide a general method for describing and developing multiscale visualizations of hierarchically structured data, designing such visualizations remains a hard and challenging problem. We use our formalism to enumerate four design patterns in the style of Gamma et al. [10] that succinctly capture the critical structure of commonly used multiscale visualizations. In addition, these patterns illustrate the use of small multiples and tables in multiscale visualizations.

Note that we are using data cubes not only because they provide a powerful mechanism for data abstraction, but also because many large and important data sets are already stored in relational databases and data cubes.

The layout of the rest of this paper is as follows. In Section 2, we survey existing approaches to multiscale visualization. Next, we describe in Section 3 how multiscale visualizations can be expressed as graphs using our Polaris formalism and data cubes and then implemented in Rivet [5]. We then present our design patterns

Research projects

- Can we build something more flexible than immens, but less complicated and more efficient than nanocubes?
- What else can we do by changing aggregation operators carefully?