

# Visualization Schemas: A User Interface Extending Relational Data Schemas for Flexible, Multiple- View Visualization of Diverse Databases

**Varun Omprakash Saini**  
vsaini@vt.edu

Thesis submitted to the faculty of  
Virginia Polytechnic Institute and State University  
in partial fulfillment of the requirements for the degree of  
Master of Science in Computer Science

Advisory Committee:  
Chris North, Chair  
Edward Fox  
Manuel A. Pérez-Quiñones

April 28<sup>th</sup> 2003  
Blacksburg, Virginia

**Keywords:** Datafaces, Visualization Schema, Information Visualization, Joinpath

# Visualization Schemas: A User Interface Extending Relational Data Schemas for Flexible, Multiple-View Visualization of Diverse Databases

by Varun Saini

## Abstract

Information visualizations utilizing multiple coordinated views allow users to rapidly explore complex data spaces and discover complex relationships. Most multiple-view visualizations are static with regard to the views that they provide and the coordinations that they support. Despite significant progress in the field of Information Visualization and development of novel interaction techniques, user interfaces for exploring data have lacked flexibility. As a result, the vast quantities of information rapidly being collected in databases are underutilized and opportunities for advancement of knowledge are lost. This research proposes the central concept of *visualization schemas* based on the Snap-Together Visualization (Snap) model, analogous to the successful database concept of data schemas, which will enable dynamic specification of information visualizations for any given database without programming.

Relational databases provide significant flexibility to organize, store, and manipulate an infinite variety of complex data collections. This flexibility is enabled by the concept of relational data schemas, which allow data owners to easily design custom databases according to their unique needs. We bring the same level of flexibility to visualization development through visualization schemas. *Visualization Schemas* is a conceptual model, user interface, and software architecture while *Fusion* is the implemented system that enable users to rapidly and dynamically construct personalized multi-view visualization workspaces by coordinating visualizations in ways unforeseen by the original developers.

## Acknowledgements

Firstly, I would like to thank Dr. North for his support and encouragement. You have been the source of constant motivation and guidance. You have filled me with interest and enthusiasm towards the emerging field of information visualization.

Thanks to Nathan Conklin and Sanjini Jayraman. You both have been a great source of inspiration for me. Nathan, your support and appreciation has played a major role in my success. You always motivated and helped me in achieving everything I wanted.

I thank all those who participated with me on various project teams to make this work possible. Thanks to Rohit, Atul, and Jeevak for working with me on Snap. Thanks to Umer and Aniket for helping in creating a sound architecture for Fusion. Thanks to those who worked with me on Information Visualization team. Thanks to Shalini, Rohan, Amit and Candida for working with me on visualization schemas.

All my work wouldn't have been possible without the contribution of my fellow developers. I thank Kiran and Qiang for development efforts. Thanks to my VTRUCS's companions: Amit Nithian, Craig Sinning, Chris Shirk, Larry Leventhal, David Longley, and Scott Walker.

Finally, I wish good luck to all current and future members of the Fusion team.

My success belongs to my parents whose love and affection has given me the strength to dream and to achieve. I also thank Shalini for the immense support she provided. It's her belief in me and my abilities that has kept me going. Shalini, you are the reason for my undying desire to excel.

## Table of Contents

<b>ABSTRACT .....</b>	<b>II</b>
<b>ACKNOWLEDGEMENTS .....</b>	<b>III</b>
<b>CHAPTER 1 VISUALIZATION SCHEMA : INTRODUCTION AND MOTIVATION.....</b>	<b>1</b>
1.1 PROBLEM.....	1
1.1.1 Proliferation of Data.....	1
1.1.2 Multi-view visualizations.....	1
1.1.3 Multi-view visualizations are Hard to build and maintain.....	2
1.1.3.1 Tradeoff.....	3
1.1.4 Need Better ways to construct multi-view visualizations.....	4
1.2 PREVIOUS SOLUTION: SNAP-TOGETHER VISUALIZATION.....	5
1.2.1 Previous Snap Model.....	5
1.2.2 Previous User Interface.....	6
1.2.3 Previous Snap Architecture.....	6
1.3 LIMITATIONS OF PREVIOUS SOLUTION.....	7
1.3.1 Model.....	7
1.3.2 User Interface.....	7
1.3.3 Architecture.....	7
1.4 RESEARCH QUESTIONS.....	8
1.5 SOLUTION – VISUALIZATION SCHEMAS.....	8
1.6 SCENARIO.....	11
1.7 CONTENT.....	17
<b>CHAPTER 2 RELATED WORK .....</b>	<b>19</b>
2.1 MULTIPLE VIEW VISUALIZATION.....	19
2.2 USER CONSTRUCTED VISUALIZATION.....	22
2.2.1 Snap-Together Visualization.....	25
2.3 RELATIONAL DATABASE SCHEMAS.....	27
<b>CHAPTER 3 VISUALIZATION SCHEMA : MODEL .....</b>	<b>31</b>
3.1 MOTIVATION.....	31
3.2 VISUALIZATION SCHEMA THEORY.....	32
3.3 COORDINATIONS AND JOINS.....	36
3.4 RELATIONSHIP BETWEEN VISUALIZATION RELATIONSHIP AND DATABASE RELATIONSHIP.....	39
3.5 DISCUSSION.....	42
3.5.1 Future Work.....	42
3.5.1.1 Data Mining.....	42
3.5.1.2 Normalization Rules.....	42
3.5.2 Example Systems.....	42
3.5.2.1 Windows Explorer.....	43
3.5.2.2 Javadocs.....	43
3.6 SUMMARY.....	44
<b>CHAPTER 4 VISUALIZATION SCHEMAS: USER INTERFACE.....</b>	<b>45</b>
4.1 OVERVIEW.....	45
4.2 VISUALIZATION SCHEMA STRUCTURE.....	46
4.2.1 Nodes.....	49
4.2.2 Edges.....	50
4.3 DATAFACES (DATABASE + INTERFACES).....	53
4.4 EVENT FEEDBACK.....	54
4.5 DATABASE SCHEMA UI.....	56
4.5.1 Database Connection Interface.....	56
4.5.2 Database Schema UI.....	56

4.5.2.1	<i>Overview view</i> .....	57
4.5.2.2	<i>Details view</i> .....	58
4.6	WORKSPACE USER INTERFACE .....	60
4.7	DISCUSSION.....	61
4.7.1	<i>Advantages</i> .....	61
4.8	FUTURE WORK AND IMPROVEMENTS.....	62
4.8.1	<i>Saving Schemas</i> .....	62
4.8.2	<i>Bookmarks</i> .....	63
4.8.3	<i>Collaboration</i> .....	64
4.8.4	<i>Feed Forward</i> .....	64
4.9	SUMMARY .....	64
<b>CHAPTER 5</b>	<b>VISUALIZATION SCHEMA : ARCHITECTURE.....</b>	<b>65</b>
5.1	OVERVIEW.....	65
5.1.1	<i>Previous Snap architecture</i> .....	66
5.2	COORDINATION MANAGER.....	67
5.2.1	<i>Coordination Graph</i> .....	67
5.3	INTEGRATED MULTIPLE DATA SOURCES.....	69
5.4	DATABASE SCHEMA.....	70
5.4.1	<i>Relational database schema structure</i> .....	70
5.4.2	<i>Virtual Database Schema</i> .....	71
5.5	EVENT TRANSLATION & PROPAGATION.....	73
5.5.1	<i>Key Converter</i> .....	74
5.6	DISCUSSION.....	76
5.6.1	<i>Visualization Schema for constructing Data Driven Web sites</i> .....	76
5.6.2	<i>JavaScript Adapter</i> .....	77
5.6.3	<i>Performance issues</i> .....	77
5.6.3.1	<i>Key conversion</i> .....	77
5.6.3.2	<i>Schema retrieval</i> .....	77
5.7	SUMMARY .....	77
<b>CHAPTER 6</b>	<b>CONCLUSIONS .....</b>	<b>79</b>
6.1	CONTRIBUTION.....	79
6.2	BENEFITS.....	80
6.3	LIMITATIONS AND FUTURE WORK.....	81
6.4	CONCLUSION.....	82
<b>APPENDIX A: SCENARIOS .....</b>	<b>.....</b>	<b>83</b>
US CENSUS.....	.....	83
WEB LOGS .....	.....	84
FILES AND FOLDERS.....	.....	86
<b>REFERENCES .....</b>	<b>.....</b>	<b>88</b>

## Table of Figures

FIGURE 1: DYNAMAPS SHOWING USE OF MULTIPLE VIEWS FOR ANALYSIS OF CENSUS DATA. SELECTING STATES WITH HIGH VALUE OF EDUCATIONAL ATTAINMENT AND INCOME HIGHLIGHTS THE SAME STATES IN MAP VIEW. .... 2

FIGURE 2: CENSUS BUREAU FACT FINDER. USERS TYPICALLY SEARCH FOR ONE PARTICULAR VALUE AT A TIME. THIS SIMPLE INTERFACE IS FASTER TO BUILD BUT IS NOT AS POWERFUL AS DYNAMAPS (FIGURE 1). .... 3

FIGURE 3: (A) PLOT SHOWING SPEED OF DEVELOPMENT AND STRENGTH OF VISUALIZATION IN FINDING TRENDS AND RELATIONSHIPS. *DYNAMAPS* IS POWERFUL BUT TAKES TIME TO BUILD WHEREAS *FACT FINDER* IS FAST TO BUILD BUT IS NOT AS POWERFUL AS *DYNAMAPS*. (B) PLOT SHOWING A SIMILAR TRADEOFF IN THE DATABASE WORLD. DATABASES ARE POWERFUL BUT IT TAKES TIME TO BUILD THEM. TEXT FILES ARE FASTER TO BUILD BUT ARE WEAK AND DO NOT SUPPORT ADVANCED USER ACTIONS (E.G. QUERIES). .... 4

FIGURE 4 – OVERVIEW OF THE SNAP COORDINATION MODEL. PLOT ENCAPSULATES THE FOLDERS RELATION AND A TABULAR VISUALIZATION ENCAPSULATES THE FILES. A SELECT EVENT WOULD OCCUR IN THE PLOT WITH THE PRIMARY KEY OF THE FOLDER SELECTED. .... 5

FIGURE 5 – PREVIOUS SNAP VISUALIZATION MENU USED TO CHOOSE A RELATION TO LOAD INTO VISUALIZATION. .... 6

FIGURE 6 – PREVIOUS SNAP SPECIFICATION DIALOG USED TO EDIT THE LIST OF COORDINATIONS. .... 6

FIGURE 7: VISUALIZATION SCHEMAS BUSTS THE TRADEOFF OF STRENGTH VS. SPEED OF DEVELOPMENT OF MULTI-VIEW VISUALIZATIONS. THIS IS SIMILAR TO DATABASE SCHEMAS BUSTING THE TRADEOFF OF STRENGTH VS. SPEED OF DEVELOPMENT FOR DATA STORAGE MECHANISMS IN THE DATABASE REALM. .... 9

FIGURE 8 (ALSO FIGURE 41) : VISUALIZATION SCHEMAS AND ITS RELATIONSHIP TO DATABASE SCHEMA. DATABASE SCHEMA BUILDS ON TOP OF DATABASES AND VISUALIZATION SCHEMA BUILDS ON TOP OF DATABASE SCHEMA. .... 10

FIGURE 9 – A USER CAN CONNECT SNAP TO A NETWORK OR LOCAL DATABASE IN ORDER TO BEGIN VISUALIZATION CONSTRUCTION. THE LEFT FRAME SHOWS THE SNAP-CONFIGURATION PANEL AND THE RIGHT SHOWS FUSION DOCUMENTATION. .... 12

FIGURE 10 – USERS CAN CONFIGURE THE DATABASE SCHEMA. THEY CAN ADD AND DELETE NEW RELATIONSHIPS AND RELATIONS. THESE CHANGES AFFECT ONLY FUSION’S VIEW OF THE SCHEMA AND NOT THE ACTUAL UNDERLYING DATABASE SCHEMA. .... 13

FIGURE 11 – USERS CAN ORGANIZE THE FRAME LAYOUT AND SELECT COMPONENTS ONCE SNAP IS CONNECTED TO A DATABASE. .... 13

FIGURE 12 – A USER CHOOSES THE GEOGRAPHIC MAP TO SEE STATES DATA. USER SPECIFIES THE DATA TO BE LOADED INTO THE MAP BY DRAGGING DESIRED RELATION FROM THE DATABASE SCHEMA TO VISUALIZATION COMPONENT ICON. .... 14

FIGURE 13: USER COORDINATES THE MAP AND THE SCATTERPLOT BY CONNECTING THE SELECT PORTS OF BOTH. A SELECTION IN ONE VIEW WILL RESULT IN CORRESPONDING VALUES SELECTED IN THE OTHER VIEW. .... 15

FIGURE 14: SELECTION OF STATES WITH HIGH VALUES OF PER CAPITA INCOME AND PERCENTAGE COLLEGE GRADUATES SELECTS THESE STATES IN THE MAP SHOWING THAT THESE STATES ARE LOCATED IN THE NORTHEASTERN REGION. .... 16

FIGURE 15: FUSION COORDINATED VISUALIZATION WITH TIGHTLY COUPLED ACTIONS BETWEEN A GEOGRAPHICAL MAP, BAR CHART, AND SCATTERPLOT. VISUALIZATION WORKSPACE AND DATAFACES CONSISTING OF VISUALIZATION SCHEMA AND DATA SCHEMA IS ALSO SHOWN. .... 17

FIGURE 16: *FUSION* SYSTEM DIAGRAM SHOWING VARIOUS LAYERS OF THE SYSTEM. HORIZONTAL ROWS SHOW THE SYSTEM LAYERS WHEREAS VERTICAL COLUMNS SHOW THE MODEL, USER INTERFACE AND ARCHITECTURE FOR EACH HORIZONTAL LAYER..... 18

FIGURE 17 – WING [44], DATASPLASH [4], HOME FINDER[2], DEVISE, SPOTFIRE, AND VISAGE ARE EXAMPLES OF HOW MULTIPLE-VIEW VISUALIZATIONS SUPPORT MANY PROBLEM DOMAINS. .... 20

FIGURE 18: VQE (A) INDICATING THE ENTIRE SET OF HOUSES, COLOR-CODED NEIGHBORHOOD AND (B) INDICATES SELECTED HOUSES HAVE BEEN DRAGGED TO VQE, WHERE THEY BECOME A DYNAMIC AGGREGATE. THE ARROWS TO THE RIGHT CONSTITUTE A MENU FOR PARALLEL NAVIGATION..... 21

FIGURE 19: DIAMOND, A PROTOTYPE SYSTEM FOR INTERACTIVE EXPLORATION OF MULTIDIMENSIONAL DATA [35].... 22

FIGURE 20 - AVS IS A MULTIPLE-VIEW VISUALIZATION SYSTEM THAT UTILIZES THE DATAFLOW MODEL..... 22

FIGURE 21 – A DATAFLOW NETWORK WITH COMPUTATIONAL COMPONENTS AS NODES THAT CONTAIN INPUT AND/OR OUTPUT PORTS. LINKS ARE USED TO CONNECT THESE PORTS AND DATA FLOWS DOWNSTREAM (TOP-TO-BOTTOM). .... 23

FIGURE 22 - SIEVE'S DATAFLOW MODEL REQUIRES PROGRAMMED CONNECTIONS. .... 23

FIGURE 23: GEOVISTA DESIGN WINDOW ALLOWS USERS TO CREATE MULTI-VIEW VISUALIZATION BY DEFINING CONNECTIONS BETWEEN COMPONENTS. A COORDINATION IS DRAWN BETWEEN COLOR CODED CONNECTORS.... 24

FIGURE 24 : JAVA'S BDK. BEAN EVENTS CAN BE TIED TO EACH OTHER TO BUILD A COMPONENT BASED APPLICATION. BDK ADOPTS A FORM BASED APPROACH FOR SPECIFICATION OF EVENTS FOR INDIVIDUAL COMPONENTS. .... 24

FIGURE 25: ISYS [25]. SHOWING MULTIPLE RELATED VIEWS SPECIALIZED FOR BIOINFORMATICS DATA. ISYS PROVIDES DYNAMIC, FLEXIBLE PLATFORM FOR THE INTEGRATION OF BIOINFORMATICS SOFTWARE TOOLS. .... 25

FIGURE 26 - SNAP-TOGETHER VISUALIZATION ALLOWS COORDINATION BASED ON RELATIONAL JOINS. .... 26

FIGURE 27 - THE COORDINATION PROPERTIES WINDOW ALLOWS FOR THE SPECIFICATION OF A COORDINATION..... 27

FIGURE 28 – THE VIEW PROPERTIES WINDOW ALLOWS FOR THE CONFIGURATION OF THE DATA AND THE VISUALIZATION. .... 27

FIGURE 29: DIAGRAM SHOWING THE VISUAL REPRESENTATION OF THE QUERY IN VOODOO..... 28

FIGURE 30: ASSOCIATIONS VIEW IN SEEDATA. IT SHOWS THE ATTRIBUTES IN A CURRENT RELATION AND THE ASSOCIATED RELATIONS GIVEN IN DATABASE SPECIFICATIONS. ASSOCIATIONS ARE DIRECTED. THE POSITION OF THE OTHER RELATIONS IN THE WINDOW INDICATES THEIR ASSOCIATION WITH THE CURRENT RELATION..... 28

FIGURE 31: DIAGRAMMATIC DATA SCHEMA FOR A DATABASE ..... 29

FIGURE 32: THE ELEMENTS OF THE RMM DATA MODEL ..... 30

FIGURE 33: RMD DIAGRAM FOR *FACULTY-COURSE* WEBSITE NAVIGATION DESIGN..... 30

FIGURE 34 – AN EXAMPLE DATA SCHEMA FOR A DATABASE OF HITS TO OUR WEBSITE. “URLS” STORES INFORMATION ABOUT PAGES ON OUR WEBSITE. “REFERRERS” STORES INFORMATION ABOUT EXTERNAL WEBSITES THAT HAVE

LINKS TO OUR WEBSITE. “HITS” STORES INFORMATION ABOUT EACH HIT , INCLUDING A REFERENCE TO THE PAGE REQUESTED IN “URLS” AND THE EXTERNAL REFERRING SITE IN “REFERRERS” .....	36
FIGURE 35 – AN EXAMPLE MULTIPLE-VIEW VISUALIZATION CONSTRUCTED WITH SNAP FOR THE DATABASE SHOWN IN FIGURE 34. THE WEBSITE MAP GENERATED FROM THE URLS IS SHOWN IN THE TREEVIEW (TOP LEFT). SELECTING A PAGE IN THE MAP DISPLAYS THE PAGE IN THE WEB BROW SER (TOP RIGHT ), AND DISPLAYS THE DISTRIBUTION OF HITSTO THAT PAGE IN THE SCATTER PLOT (TOP CENTER). SELECTING PAGES ALSO HIGHLIGHTS REFERRING SITES LISTED IN THE TABLE VIEW (BOTTOM LEFT) . LIKewise, SELECTING REFERRING SITES HIGHLIGHTS PAGES LINKED TO, AND SHOWS THEIR HIT S IN THE PLOT . CLICKING A REFERRER SHOWS ITS PAGE IN THE OTHER WEB BROWSER (BOTTOM RIGHT ).....	37
FIGURE 36: BRUSHING AND LINKING ACROSS SCATTER PLOTS ENCAPSULATING STATES DATA INDICATES THAT STATES HAVING LOW PERCENTAGE OF COLLEGE AND HIGH SCHOOL GRADS (LEFT ) HAVE LOW PER CAPITA INCOME (RIGHT).....	40
FIGURE 37: OVERVIEW AND DETAIL WITH WEB FRAMES.....	40
FIGURE 38: SYNCHRONIZED SCROLLING IN <i>COMPARE IT!</i> , A FILE COMPARE UTILITY [14]. .....	41
FIGURE 39: MICROSOFT WINDOWS EXPLORER DISPLAYING TWO VIEWS. THE TREE-VIEW SHOWING FOLDERS ON THE LEFT AND THE DETAILS-VIEW ON THE RIGHT .....	43
FIGURE 40: JAVADOCs FOR JAVA SDE 1.4.1. THE LEFT TOP FRAME SHOWS THE PACKAGE VIEW , THE BOTTOM LEFT VIEW SHOWS THE CLASS VIEW. THE RIGHT FRAME SHOWS THE DETAILS VIEW FOR THE CLASSES. ....	44
FIGURE 41: VISUALIZATION SCHEMAS AND IT'S RELATIONSHIP TO DATABASE SCHEMA. DATABASE SCHEMA BUILDS ON TOP OF DATABASES AND VISUALIZATION SCHEMA BUILDS ON TOP OF DATABASE SCHEMA.....	46
FIGURE 42: ( ALSO FIGURE 34) AN EXAMPLE DATA SCHEMA FOR A DATABASE OF HTTP HITS TO OUR WEBSITE. “URLS” STORES INFORMATION ABOUT PAGES ON OUR WEBSITE. “REFERRERS” STORES INFORMATION ABOUT EXTERNAL WEBSITES THAT HAVE LINKS TO OUR WEBSITE. “HITS” STORES INFORMATION ABOUT EACH HIT , INCLUDING A REFERENCE TO THE PAGE REQUESTED IN “URLS” AND THE EXTERNAL REFERRING SITE IN “REFERRERS” . .....	47
FIGURE 43: VISUALIZATION SCHEMA FOR THE MULTIPLE-VIEW VISUALIZATION SHOWN IN FIGURE 44.....	48
FIGURE 44: (ALSO FIGURE 35) AN EXAMPLE MULTIPLE-VIEW VISUALIZATION CONSTRUCTED WITH SNAP FOR THE DATABASE SHOWN IN FIGURE 42. THE WEBSITE MAP GENERATED FROM THE URLS IS SHOWN IN THE TREEVIEW (TOP LEFT) . SELECTING A PAGE IN THE MAP DISPLAYS THE PAGE IN THE WEB BROWSER (TOP RIGHT ), AND DISPLAYS THE DISTRIBUTION OF HITS TO THAT PAGE IN THE SCATTER PLOT (TOP CENTER). SELECTING PAGES ALSO HIGHLIGHTS REFERRING SITES LISTED IN THE GRAPHICAL-TABLE VIEW (BOTTOM LEFT) . LIKewise, SELECTING REFERRING SITES HIGHLIGHTS PAGES LINKED TO, AND SHOWS THEIR HIT S IN THE PLOT . CLICKING A REFERRER SHOWS ITS P AGE IN THE OTHER WEB BROWSER (BOTTOM RIGHT ) . .....	48
FIGURE 45: A NODE IN THE VISUALIZATION SCHEMA REPRESENTS AN INSTANTIATED VISUALIZATION COMPONENT ...	49
FIGURE 46: AN EDGE IN THE VISUALIZATION SCHEMA REPRESENTS A COORDINATION BETWEEN VISUALIZATION COMPONENTS. A COORDINATION BETWEEN THE URLS AND REFERRERS RELATIONS OFFERS A LIST OF TWO ALTERNATIVE MANY-TO-MANY JOINS. ....	51

FIGURE 47: PORT MOVING AND SPLITTING. (A) SHOWS INITIAL POSITION OF PORTS. (B) PORTS MOVE SO THAT THEY ARE ON THE CLOSEST POSSIBLE SIDE. (C) PORTS SPLIT TO PREVENT THE GRAPH FROM BE COMING CLUTTERED..... 51

FIGURE 48: DATAFACES: SHOWING THAT BOTH THE PARALLEL-COORDINATES AND THE TREEMAP [39] ENCAPSULATE THE REQUESTS RELATION..... 53

FIGURE 49: DATAFACES DISPLAYING A JOINPATH IN THE DATABASE SCHEMA CORRESPONDING TO A COORDINATION SELECTED IN THE VISUALIZATION SCHEMA..... 54

FIGURE 50: VISUALIZATION SCHEMA EVENT FEEDBACK SHOWING THE RESULT OF A “SELECT” ACTION ON THE TABLE VISUALIZATION COMPONENT ENCAPSULATING STATES RELATION (LEFT). TWO COORDINATIONS (COLORED) WERE USED FOR EVENT PROPAGATION. THE ORDER OF COORDINATION PROCESSING IS INDICATED NEXT TO THE COORDINATION..... 55

FIGURE 51: VISUALIZATION SCHEMA EVENT FEEDBACK SHOWING CYCLE RESOLUTION AS A RESULT OF “SELECT” ACTION ON THE TABLE VISUALIZATION COMPONENT ENCAPSULATING STATES RELATION (TOP LEFT). THREE COORDINATIONS (COLORED) WERE USED FOR EVENT PROPAGATION. THE ORDER OF COORDINATION PROCESSING IS INDICATED NEXT TO THE COORDINATION..... 55

FIGURE 52: DATABASE CONNECTION USER INTERFACE. USER CAN CONNECT TO MULTIPLE REMOTE OR LOCAL DATABASES SIMULTANEOUSLY..... 56

FIGURE 53: DATABASE SCHEMA OVERVIEW-VIEW. (A)THE ATTRIBUTE LIST SHOWS THE ATTRIBUTES FOR THE SELECT TABLE, I.E., URLS.(B) DATABASE SCHEMA WITH ATTRIBUTE LIST MINIMIZED/HIDDEN..... 58

FIGURE 54: DATABASE SCHEMA-DETAILS VIEW. EDITABLE DATABASE SCHEMA AT THE RIGHT AND LIST OF DELETED AND AVAILABLE TABLES ON THE LEFT .DATABASE SCHEMAS FOR TWO SEPARATE DATABASES (*CENSUS AND WEBLOGS*) ARE SHOWN ON THE RIGHT ..... 59

FIGURE 55: SQL DIRECT . SUPPORTS RUNNING SQL QUERIES DIRECTLY AGAINST THE UNDERLYING DATABASES. ADVANCED USERS CAN USE THIS FOR CREATING TEMPORARY TABLES AND VIEWS..... 59

FIGURE 56: WORKSPACE WINDOW MANAGEMENT TASK BAR. THIS APPEARS BELOW EVERY VISUALIZATION COMPONENT IN THE VISUALIZATION WORKSPACE ..... 60

FIGURE 57: FRAME SPLITTING. (A) SHOWS A *TABLE* LOADED IN A FRAME. (B) SHOWS THE SAME *TABLE* ON THE RIGHT HAND SIDE AFTER SPLITTING. A BLANK FRAME IS CREATED ON THE LEFT SIDE WHICH DISPLAYS THE VISUALIZATION MENU ..... 61

FIGURE 58: SAMPLE XML FILE FOR SAVING AND RESTORING SCHEMA INFORMATION. INFORMATION IS STORED IN THREE DISTINCT SECTIONS (A) DATABASE CONNECTION INFORMATION, (B) WORKSPACE AND LAYOUT INFORMATION, AND (C) RELATIONSHIP BETWEEN VISUALIZATION COMPONENTS..... 63

FIGURE 59 – WEB-BASED SNAP ARCHITECTURE [9] INDICATING THE SEPARATE LAYERS NEEDED TO IMPLEMENT THE SNAP SYSTEM. THE DATA SOURCE MAINTAINS THE DATA TO BE VISUALIZED. THE FIRST LAYER DEALS WITH PROVIDING CONNECTIVITY TO DATA SOURCES AND DESCRIBES JOIN ASSOCIATIONS BETWEEN THE SEPARATE DATA RELATIONS. THE SECOND LAYER SUPPORTS THE ASSIGNMENT OF DATA RELATIONS TO VISUALIZATION COMPONENTS, AND THE COORDINATION OF EVENTS BETWEEN COMPONENTS. THE THIRD LAYER HANDLES ALL COMMUNICATION WITH THE VISUALIZATION COMPONENTS, AND IS RESPONSIBLE FOR THE RECEIVING AND

FIRING OF EVENTS. COORDINATION MANAGER, COORDINATION GRAPH, AND DATABASE SCHEMA WERE CHANGED TO ADD SUPPORT FOR MULTIPLE DATABASES AND EVENT PROPAGATION ACROSS COMPOUND JOINS .....	66
FIGURE 60 – COORDINATION GRAPH CLASS DIAGRAM.....	67
FIGURE 61: EXAMPLE COORDINATION GRAPH WITH FOUR COORDINATED VISUALIZATION COMPONENTS.....	68
FIGURE 62: COORDINATION GRAPH DATA STRUCTURE SHOWING COORDINATION NODES FOR EACH OF THE VISUALIZATIONS REPRESENTED IN FIGURE 61. EACH NODE STORES A HASH TABLE WITH ACTIONS AS HASH KEYS. ....	68
FIGURE 63: DATABASE MANAGER USES DATA SOURCE INFO, SCHEMA GRAPH CONNECTION MANAGER. SCHEMA GRAPH USES RELATIONSHIP NODE TO STORE RELATIONSHIPS BETWEEN RELATIONS. CONNECTION MANAGER MANAGES VARIOUS DATABASE CONNECTIONS WHILE DATA SOURCE INFO STORES DATABASE SPECIFIC INFORMATION FOR DIFFERENT TYPE OF DATABASES.....	70
FIGURE 64: DATABASE SCHEMA INTERFACE TO THE STRUCTURE .....	71
FIGURE 65: DATABASE MANAGER INTERFACE TO ACCESS DATA. TRANSLATE EVENT METHOD CONVERTS THE INPUT KEYS FROM A SOURCE TABLE INTO OUTPUT KEYS IN A DESTINATION TABLE USING SPECIFIED JOIN PATH. GET RESULT SET METHOD GETS A RESULTSET OF DATA FOR THE SPECIFIED QUERY.....	71
FIGURE 66: VIRTUAL RELATIONSHIP . A VIRTUAL RELATIONSHIP CAN EXIST WITHIN THE SAME DATABASE (INTRA-DATABASE) OR ACROSS DATABASES (INTER-DATABASE).....	71
FIGURE 67: VIRTUAL DATA SCHEMA BUILDS ON TOP OF DATA SCHEMA'S FOR INDIVIDUAL DATABASES. USER CAN CREATE VIRTUAL TABLES AND RELATIONSHIPS IN THE VIRTUAL SCHEMA.....	72
FIGURE 68: VIRTUAL DATA SCHEMA EDITING TOOL. USERS CAN CREATE CUSTOM DATA SCHEMA TO BE USED BY FUSION FOR COMPONENT COORDINATION AND EVENT PROPAGATION . ....	72
FIGURE 69: KEY CONVERSION. COORDINATION MANAGER USES KEY CONVERTER TO CONVERT KEYS FROM SOURCE RELATION TO DESTINATION RELATION. MAP AND TABLE ARE RELATED BY A <i>SELECT-TO-LOAD</i> RELATIONSHIP . MAP FIRES EVENTS ON <i>STATE ID</i> WHEREAS TABLE EXPECTS TO RECEIVE <i>COUNTY ID</i> . COORDINATION MANAGER CONVERTS <i>STATE ID</i> KEY VALUES TO CORRESPONDING <i>COUNTY ID</i> KEY VALUES BASED ON <i>JOINPATH</i> . ....	73
FIGURE 70: KEY CONVERTER. CONVERTS SOURCE RELATION PRIMARY KEYS TO DESTINATION TABLE PRIMARY KEYS BASED ON JOINPATH.....	74
FIGURE 72: DATABASE SCHEMA SHOWING ONE-TO-MANY (1: M) RELATIONSHIP BETWEEN STATES AND COUNTIES RELATIONS. ....	83
FIGURE 73: MULTI-VIEW VISUALIZATION BUILD USING FUSION AND VISUALIZATION SCHEMA FOR .....	83
FIGURE 74: DATABASE SCHEMA SHOWING THE RELATIONSHIP BETWEEN VARIOUS TABLES IN THE WEBLOGS DATABASE.....	84
FIGURE 75: MULTI-VIEW VISUALIZATION BUILD USING FUSION AND VISUALIZATION SCHEMA FOR WEBLOGS DATABASE (FIGURE 74). DATAFACES IS SHOWN AT THE LEFT (FIGURE 76) AND VISUALIZATION WORKSPACE HAVING 5 COMPONENTS IS SHOWN AT THE RIGHT (FIGURE 77). ....	84
FIGURE 76: DATAFACES FOR THE VISUALIZATION CONSTRUCTED IN FIGURE 75 .....	85
FIGURE 77: VISUALIZATION WORKSPACE SHOWING 5 COORDINATED COMPONENTS. ....	85

FIGURE 78: DATABASE SCHEMA FOR FILES AND FOLDERS. SHOWING A ONE TO MANY RELATIONSHIP BETWEEN  
 FOLDERS AND FILES. .... 86

FIGURE 79: DATAFACES SHOWING VISUALIZATION SCHEMA TILES ABOVE THE DATASHEMA OVERVIEW. .... 86

FIGURE 80: FUSION USED AS A HARD DISK EXPLORATION TOOL. THE TREE-VIEW INDICATED THE FOLDERS HIERARCHY.  
 THE BAR CHART DISPLAYS THE CONTENTS BY SIZE. THE WEB PAGE LOADS THE CONTENTS. ABOVE SCENARIO  
 DISPLAYS THAT THE LARGEST IMAGE FILE IN THE SNAP DIRECTORY I.E. FUSION.BMP (LOADED IN BOTTOM VIEW)  
 ..... 87

# Chapter 1 Visualization Schema: Introduction and Motivation

## 1.1 Problem

### 1.1.1 Proliferation of Data

Advances in database technology have enabled the widespread collection and proliferation of data. Almost all organizations maintain databases of relevant information. For example, Census bureau collects vast quantity of data regarding population distribution, economic status, educational status, etc. for all states. Data-driven websites, customer information, experimental results, historical trends, production information, etc. are some other examples of use of databases. The advent of relational databases and database schemas has played a significant role in the widespread collection and proliferation of data. Data schemas enable database administrators to design and define custom database instances that satisfy their unique needs. Typically these database instances have multiple relations and are spread across multiple servers. A database administrator enjoys high flexibility, supported by the relational database schemas, in organizing and maintaining the data. A visualization or user interface designer, on the other hand, does not enjoy this flexibility. Visualization designers struggle to keep pace with the rate of data schema change. Interfaces and Visualizations frequently need to be updated either because of schema change or because of changes in user tasks.

### 1.1.2 Multi-view visualizations

Single view visualizations (*visualization components*) are weak when data with multiple attributes needs to be explored. They provide the user with a single representation of data, typically concentrating on one or two attributes. Multiple view visualizations are typically used for data having multiple attributes. For example, DynaMaps (Figure 1), a multi-view visualization, is used by the census bureau to analyze the data they collect. A Multi-view visualization shows two or more views to represent a single conceptual entity [5]. These views are typically *coordinated* so that a user action in one view results in changes in other views. Coordinated multi-view visualizations provide the user with the strength of viewing different aspects of the underlying data in different views. User can interact with any view and see the

changes in other coordinated views. Each view provides a different perspective of the data to the user.

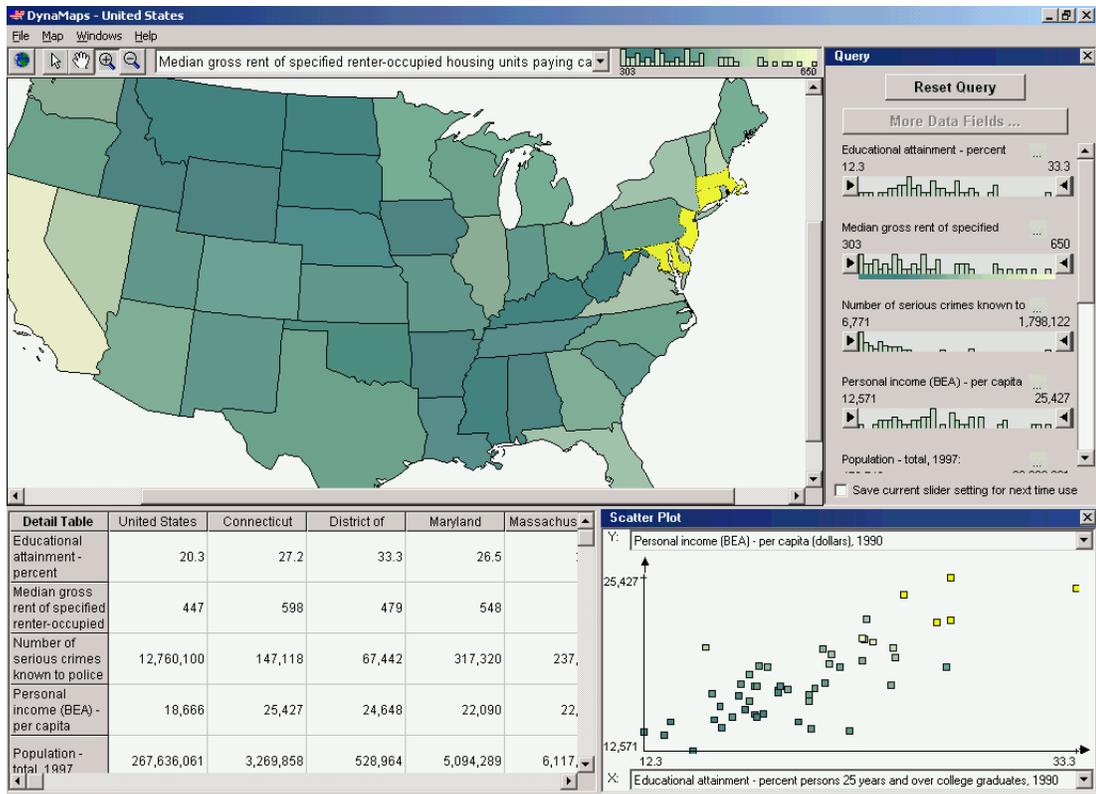


Figure 1: DynaMaps showing use of multiple views for analysis of census data. selecting states with high value of educational attainment and income highlights the same states in map view.

### 1.1.3 Multi-view visualizations are Hard to build and maintain

Multi-view visualizations are more powerful than single view visualization but are generally harder to build. As explained by [27], the set of visualizations and coordinations needed in any given situation depends heavily on:

- **data:** different data sets have different features and structure.
- **tasks:** what does the user want to accomplish with the data?
- **users:** there is tremendous variation between users in individual user preferences, experience levels, etc. For example, while Windows Explorer is helpful for some users and tasks, system administrators may need alternate visualizations.

A unique visualization is needed to support distinct user tasks on distinct database schemas and hence the number of needed combinations of visualizations and coordinations explodes exponentially, and implementation becomes intractable. Moreover, Visualizations become unusable as soon as either the user task or the database schema or both change. A change in database schema can result in significant redesign and reimplementation efforts on the part of developers, who often cannot keep pace with the rate of schema or task change.

A simple single view visualization, on the other hand, can be build in short time without much effort but it lacks the strength for exploring unforeseen relationships. These visualizations support only the most common features covering many tasks and do not utilize many of the powerful visualization techniques developed by researchers in the field [31]. As a result, database users are given only the simplest of user interfaces to data, resulting in significant limitations to data analysis. For example, Census bureau provides *Fact Finder* (Figure 2), a simple web based interface for accessing a single piece of information from the data at a time.

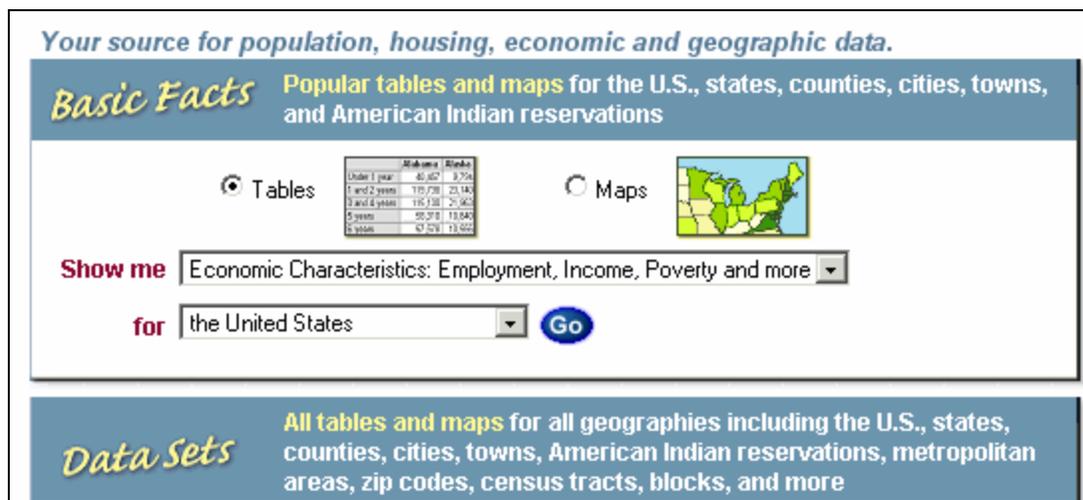


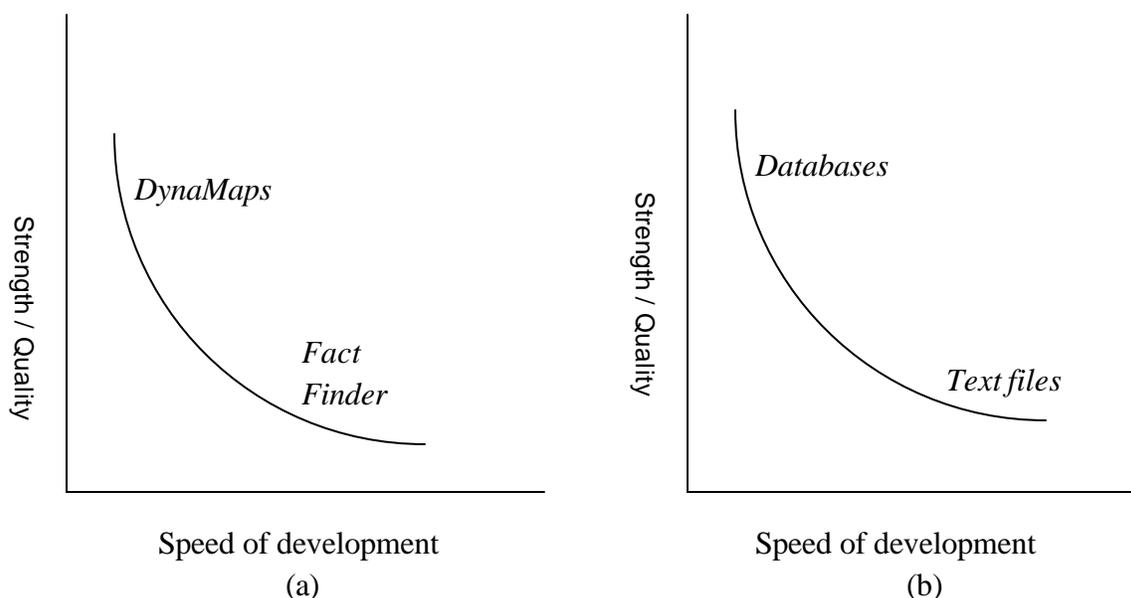
Figure 2: Census bureau Fact Finder. Users typically search for one particular value at a time. This simple interface is faster to build but is not as powerful as DynaMaps (Figure 1).

### 1.1.3.1 Tradeoff

There exists a tradeoff between the speed with which a visualization can be built and the strength of the multi-view visualizations. The development time for a simple single-view visualization

*Fact Finder* is significantly less (Figure 3) than that of *DynaMaps* but *DynaMaps* is more powerful in finding underlying relationships and trends in data.

Before visual database schemas, a similar tradeoff existed in the database world where it was possible to store data in a simple formats (e.g. text files) quickly, but these formats did not support advanced user actions (queries etc.) and were not maintainable. Databases, on the other hand, provided support for advanced user actions and were maintainable but were not quick to build. Advanced users were required for creating databases and storing data in appropriate format.



**Figure 3: (a) Plot showing speed of development and strength of visualization in finding trends and relationships. *DynaMaps* is powerful but takes time to build whereas *Fact Finder* is fast to build but is not as powerful as *DynaMaps*. (b) Plot showing a similar tradeoff in the database world. Databases are powerful but it takes time to build them. Text files are faster to build but are weak and do not support advanced user actions (e.g. queries).**

### 1.1.4 Need Better ways to construct multi-view visualizations

Despite significant progress in the field of Information Visualization and development of new interaction techniques, user interfaces for exploring data have lacked flexibility. A unique visualization is needed to support distinct user tasks on distinct database schemas. Typically, these visualizations are custom programmed and hence lack flexibility and versatility. Support

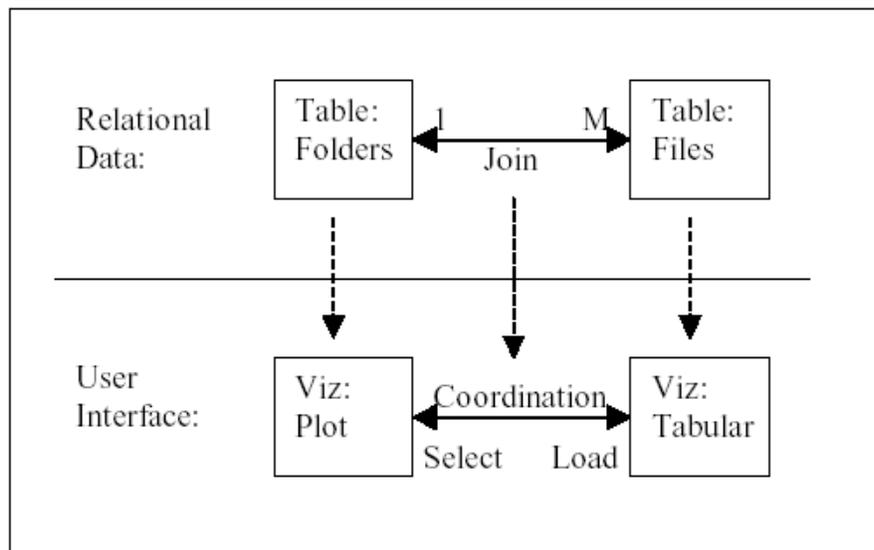
for integrating diverse data sources and diverse visualization components is critical to the success of multi-view visualizations. Custom programming is inadequate for this as the turnaround time for complete development of visualization is high. A faster approach is needed for allowing users to dynamically and rapidly assign data to views and coordinate them quickly.

## 1.2 Previous Solution: Snap-Together Visualization

North presented Snap-together visualization as a potential solution to the problem of providing flexibility in visualization development [27]. Snap enables data users to rapidly and dynamically mix and match visualizations and coordinations to construct custom exploration interfaces without programming.

### 1.2.1 Previous Snap Model

Snap Model is based on relational database model and introduces the notion that a visualization encapsulates a relation in the database [32]. As shown in Figure 4, the Plot encapsulates the Folders relation and a Tabular visualization encapsulates the Files. A *Select* event would occur in the Plot with the primary key of the folder selected. That key would be passed to the Table and it would load the associated files.



**Figure 4 – Overview of the Snap Coordination Model. Plot encapsulates the Folders relation and a Tabular visualization encapsulates the Files. A Select event would occur in the Plot with the primary key of the folder selected.**

### 1.2.2 Previous User Interface

Previous Snap user interface is aimed at supporting bottom-up construction of visualization. The data is selected before selecting the visualization. Although the user interface provides direct-manipulation support (drag-and-drop) for coordinating visualizations, it is primarily dialog based (Figure 5 and Figure 6).



Figure 5 – Previous Snap Visualization Menu used to choose a relation to load into visualization.

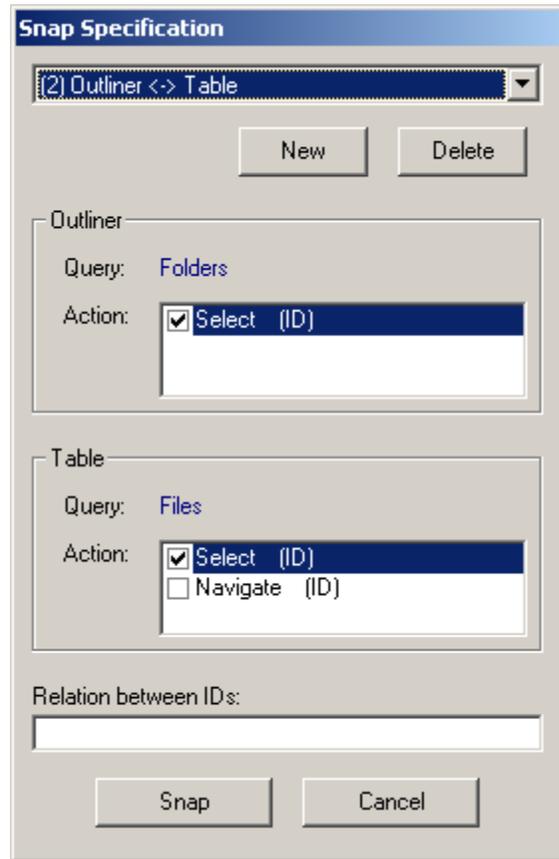


Figure 6 – Previous Snap Specification Dialog used to edit the list of coordinations.

### 1.2.3 Previous Snap Architecture

Previous Snap version is implemented only for the Windows platform. It utilizes Microsoft COM/ActiveX model for communication in the API. Access to ODBC databases is accomplished through Microsoft DAO object model. [27] explains the Snap architecture extensively.

## **1.3 Limitations of Previous Solution**

### **1.3.1 Model**

*Multiple tuple actions:* Previous Snap model supports only single tuple actions. For example, users can only select one item in a visualization component hence limiting visualizations to firing events occurring on single items. Multiple tuple actions, which are common in multiple-view visualizations, are not supported.

*Use of data associations for propagating events:* Initial Snap model is cumbersome and require advanced users. It relies on the user to construct a parameterized query that would specify the association between coordinated views. Support for automatically extracting relationships from the database schema and connecting to multiple databases is lacking.

### **1.3.2 User Interface**

*Mapping to Users Mental Model:* Previous Snap user interface does not map to users mental model of coordinating multi-view visualizations effectively. It lacks enough direct-manipulation and much of the functionality is hidden behind menus.

*Coordinations were hidden:* Once the visualization design is complete, there is no visual cue of the coordination. Interaction with the views is the only means of understanding the coordination structure.

*No Overview:* No overview of the coordination structure is provided to the user. [Figure 5] was the only view displaying information about a coordination and this is a details-view. Lack of overview can significantly hinder users understanding of the coordination structure.

### **1.3.3 Architecture**

*Support for static group of Visualizations:* Previous Snap implementation is not run-time extensible. Recompiling of Snap system is needed to add new visualization components.

*Large Install Base:* Since Snap is not a client server system each component has to be installed on the local machine before it can be used with Snap.

*Multiple database support:* Users can use Snap to explore only one database at a time. This is a drawback as users can not integrate diverse data into a single visualization. Moreover, support for multiple databases is crucial in complex data analysis tasks where data from multiple sources is related.

*Data Schema Editing Support:* Users have little power to edit the database schema and define custom relationships. Such ability is crucial to exploring unforeseen relationships between seemingly unrelated data.

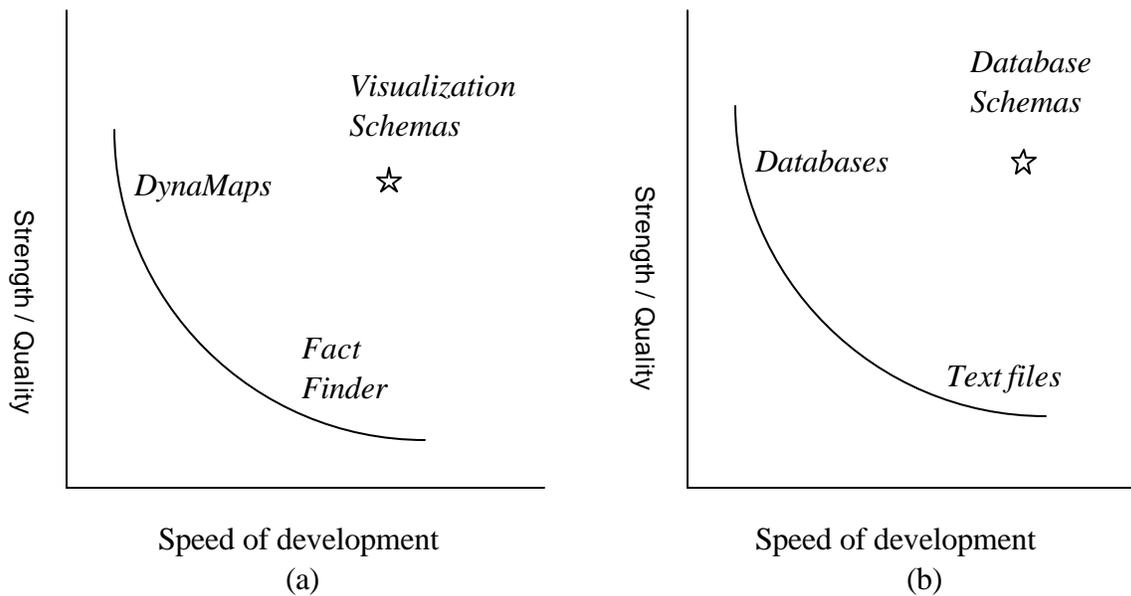
With all the above mentioned limitations, Snap falls short of solving the problem of integrating diverse data sources and diverse visualization components. Significant enhancements to Snap model and user interface are required to solve this problem.

## **1.4 Research Questions**

This research addresses the following issues: How can visualization development better match the flexibility of database construction and manipulation? How can visualization and database realms be integrated to allow efficient data analysis with the help of customized visualizations? How can users integrate diverse data sources and diverse visualization components quickly and without programming for effective construction of multi-view visualizations? The solution should be a user interface which the users can use to specify the data and visualizations to be coordinated. Other issues involve providing a theoretical foundation for specification of coordination structure. Coordinating visualizations on the basis of data that they encapsulate and underlying database join relationships is also explored.

## **1.5 Solution – Visualization Schemas**

Information visualization can learn from the success of databases. Databases provide flexibility via *data schemas* [40]. Many modern database systems provide a visual language for data schemas in the form of a diagrammatic, direct manipulation user interface.

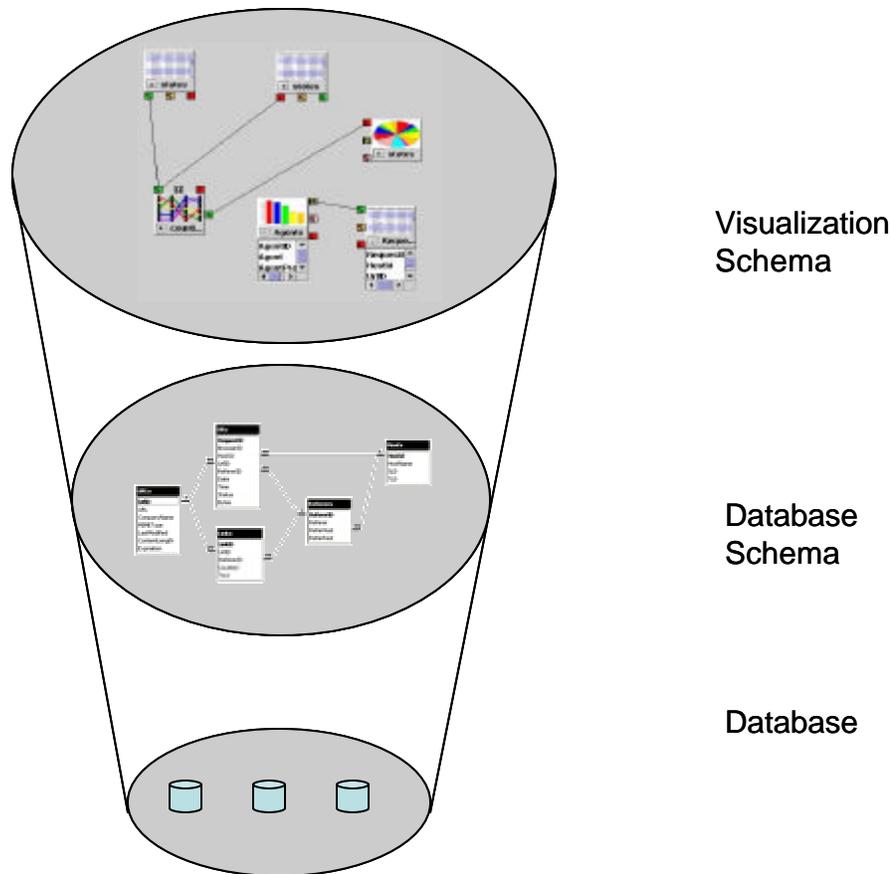


**Figure 7: Visualization Schemas busts the tradeoff of strength vs. speed of development of multi-view visualizations. This is similar to database schemas busting the tradeoff of strength vs. speed of development for data storage mechanisms in the database realm.**

Diagrammatic data schemas are generally represented as a graph. Nodes represent relations, and edges represent potential joins (primary key and foreign key relationships). Directed edges represent one-to-many relationships, and undirected edges represent one-to-one relationships.

Data schemas, especially diagrammatic data schemas, have many important benefits:

- Enables data owners to define and modify the structure of a database quickly using a simple language (Figure 7).
- Provides guidance for data design, and enforces rules of the data model (e.g. validity).
- Enables users to extract data (if queries are considered part of the schema).
- Provides an overview of database structure to help other users understand database contents.
- Enables systems to store and interpret database contents interchangeably. Data schema users have a wide range of expertise. Very complex or high performance databases may require expert data modelers, but many simpler databases are maintained by relative novices.



**Figure 8 (also Figure 41) : Visualization Schemas and its relationship to database schema. Database schema builds on top of databases and Visualization schema builds on top of database schema.**

Visualization schema is analogous to the relational database schema (Figure 8). The primary goal of visualization schemas is to enable a level of flexibility in visualization design that matches that of database design. To accomplish this, the primary guiding principle of visualization schemas is to establish a tight analogy between relational database concepts and visualization concepts. This approach has several major motivations:

- Enable close coordination between data design and visualization design, closing the gap between databases and visualization. This is important because data and visualization design often affect each other.
- Reap similar benefits to data schemas (listed previously) with visualization schemas.
- Leverage users existing knowledge of relational concepts, reducing overall learning time. Provide natural extension to relational concepts, reducing user performance time. Enable the same users to do both data and visualization tasks.

- Relational concepts provide strong data structure and organization upon which visualization design is based.
- Easily package data and visualization together for distribution.
- Relational database concepts have been very successful in the database realm. Mimicking these concepts may lead to similar success in the visualization realm.

Visualization schemas enable end-users to rapidly specify custom information visualizations for unique databases without programming (Figure 7). Visualization schemas augment existing relational data schemas to provide an integrated and coordinated approach to the design of data and visualization. *Datafaces* presents, by combining *data schema* and *visualization schema*, a coordinated user interface for simultaneous manipulation of data schema and visualization schema. *Datafaces* also makes the relationship between data schema and visualization schema apparent to the user.

*Fusion* is an implemented system that uses the concepts of visualization schema and *Datafaces* to allow users to rapidly and dynamically construct personalized visualization workspaces without programming. Users can visualize data from diverse data sources using diverse visualization components. A *visualization component* is a single view of data which can be used within a multi-view visualization.

## **1.6 Scenario**

This scenario illustrates the use of visualization schemas and *Fusion* to construct multi-view visualization in order to explore US Census data. A data analyst wants to analyze the relationship between per capita income and educational attainment. She has access to census data and would like to create a visualization to help her in analyzing the data. Simple single view visualizations are weak and would limit her analysis by restricting her to a single visual representation of data. A coordinated multi-view visualization would allow her to explore different attributes and visual representations of data. One approach is to build a custom multi-view visualization which might take a lot of time. The other approach is to use existing multi-view visualizations which are quickly available but might be inappropriate because of the specific task and schema. If she decides to use existing multi-view visualizations her analysis would be limited by the capabilities

of such a multi-view visualization and she would have to map her problem to the visualization features rather than the other way round. Fusion provides her with the power to quickly create multi-view visualizations by dynamically specifying data, visualization components, and coordinations between these components. As shown in Figure 7, Fusion allows construction of powerful and flexible visualizations in a short amount of time.

Figure 9 shows the initial Fusion screen. Fusion configuration panel is shown in the left frame while visualization workspace is shown in the right. User is presented with database connection interface. User can choose any local or remote ODBC data source like Microsoft Excel, Microsoft Access, SQL Server, Flat files, etc. to connect to. They can choose single or multiple databases. In this example, the analyst connects to remote database which publishes Census data. After successful connection to the database, Fusion retrieves the database's schema. User is then presented with an option to configure the database schema (Figure 10). User can add or delete relationships and relations.

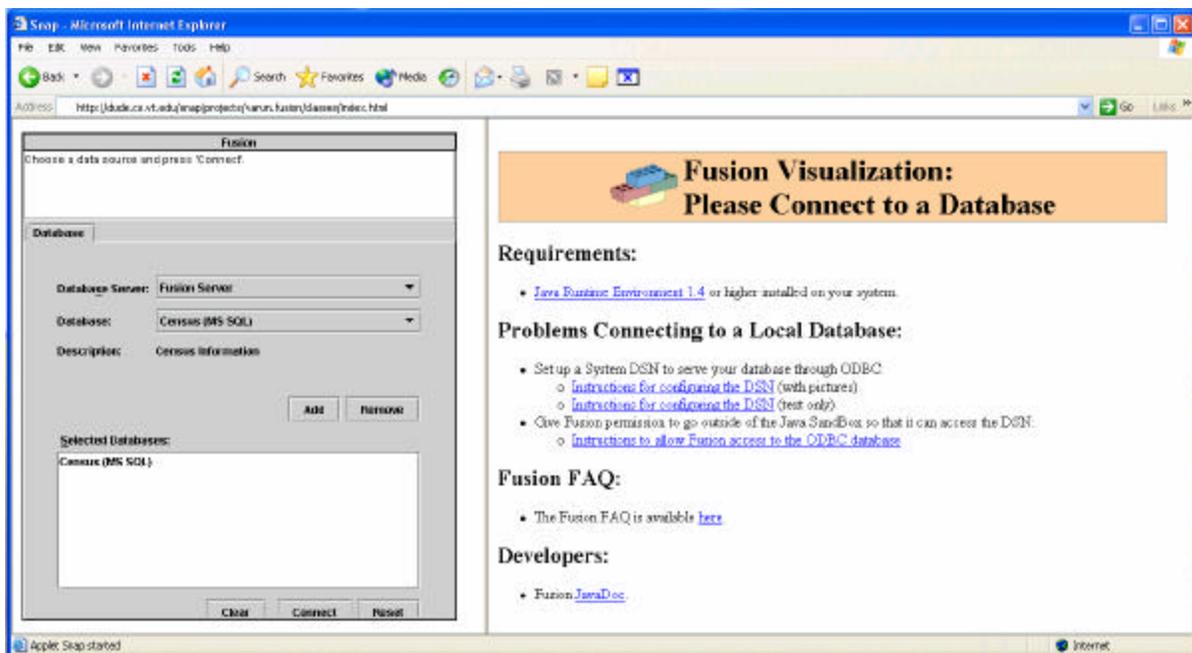


Figure 9 – A user can connect Snap to a network or local database in order to begin visualization construction. The left frame shows the Snap-configuration panel and the right shows Fusion documentation.

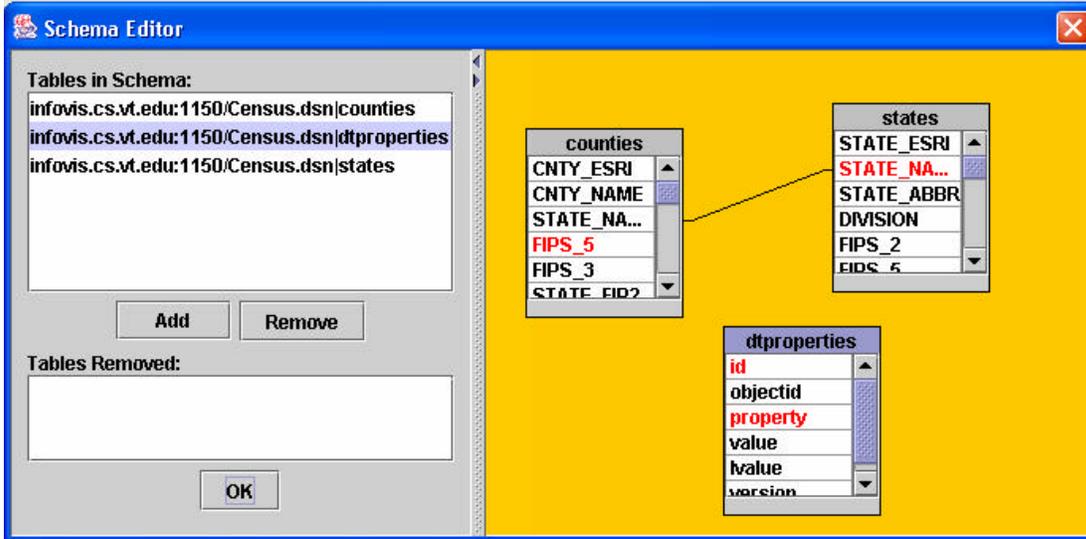


Figure 10 – Users can configure the database schema. They can add and delete new relationships and relations. These changes affect only Fusion’s view of the schema and not the actual underlying database schema.

User can arrange the individual visualization components within the workspace by recursively splitting the browser frames horizontally or vertically (Figure 11).

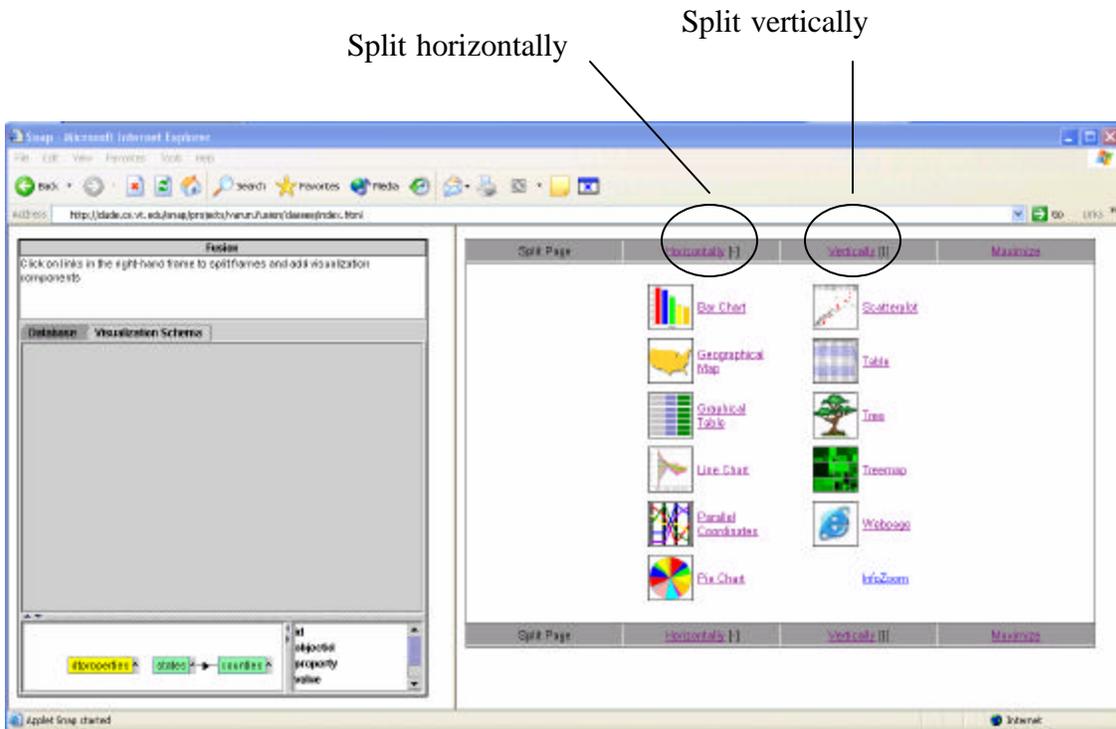
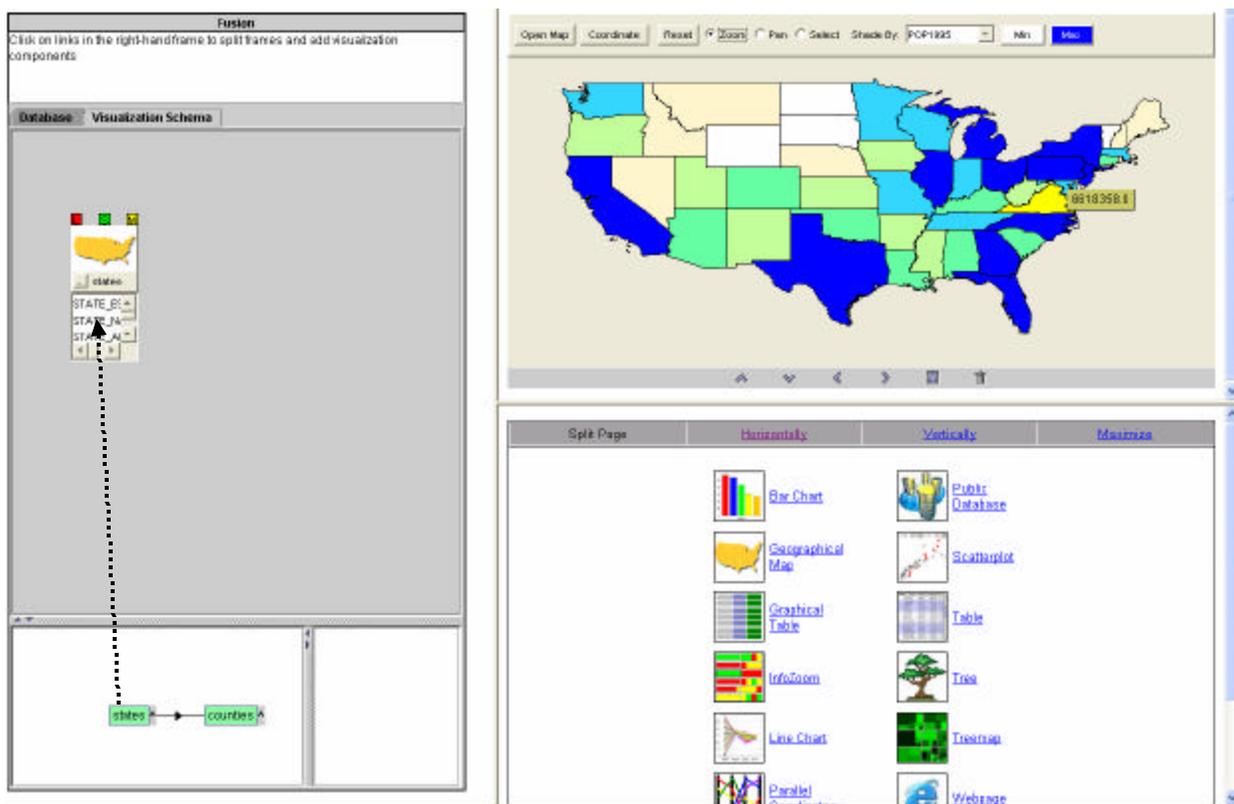


Figure 11 – Users can organize the frame layout and select components once Snap is connected to a database.

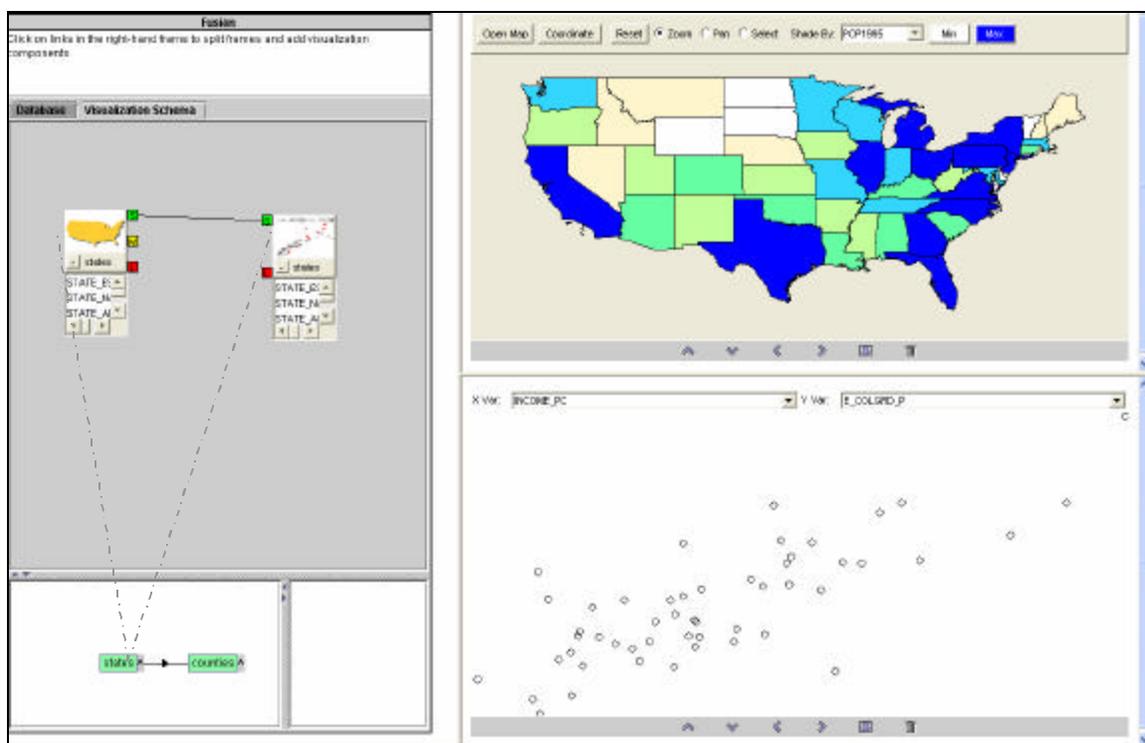
In this example, analyst splits the frame horizontally and starts by viewing state information in a *geographic map* (Figure 12). She clicks on geographic map link on the visualization page which loads a map component in the frame and a node (*visualization icon*) corresponding to the map component in the visualization schema. Data is loaded into the component by dragging the desired attributes onto the visualization icon in the visualization schema. The map can be colored by any desired data field. She chooses to color the map by population.



**Figure 12 – A user chooses the geographic map to see states data. User specifies the data to be loaded into the map by dragging desired relation from the database schema to visualization component icon.**

Within a second frame, she loads a *scatterplot* and selects the 'States' relation to load in it (Figure 13). She chooses to see the per capita income on the horizontal axis and Percentage College graduates on the vertical axis. She notices that states having higher percentage of college graduates, generally, have higher per capita income. She coordinates the map with the scatterplot by a “select-to-select” coordination which provides “brushing and linking” [6] ability i.e. selection of items in one view selects corresponding related items in another coordinated view.

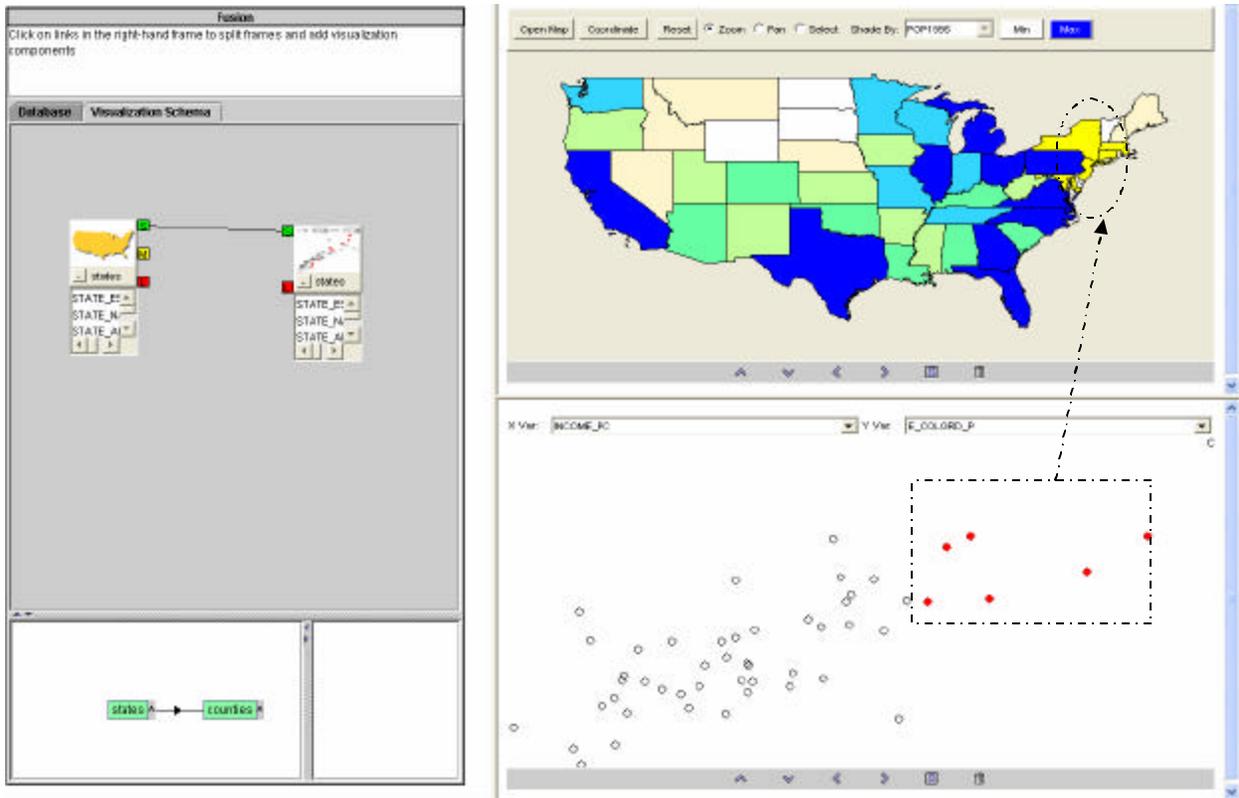
Coordinations are created by connecting appropriate ports in visualization icons by dragging a link from one port to the other. A *Port* in the visualization icon represent an action supported by the corresponding visualization. She selects the states with high value of percentage college graduates and per capita income in the scatterplot and this selects the same states in the map (Figure 14). She observes that these states are located in the north eastern region. Such relationships and observations are typically very difficult to find using a single view. The choice of view and coordination is important in finding unforeseen relationships and hence flexibility in choosing view and coordinations is paramount to the success of data analysis.



**Figure 13: User coordinates the map and the scatterplot by connecting the select ports of both. A selection in one view will result in corresponding values selected in the other view.**

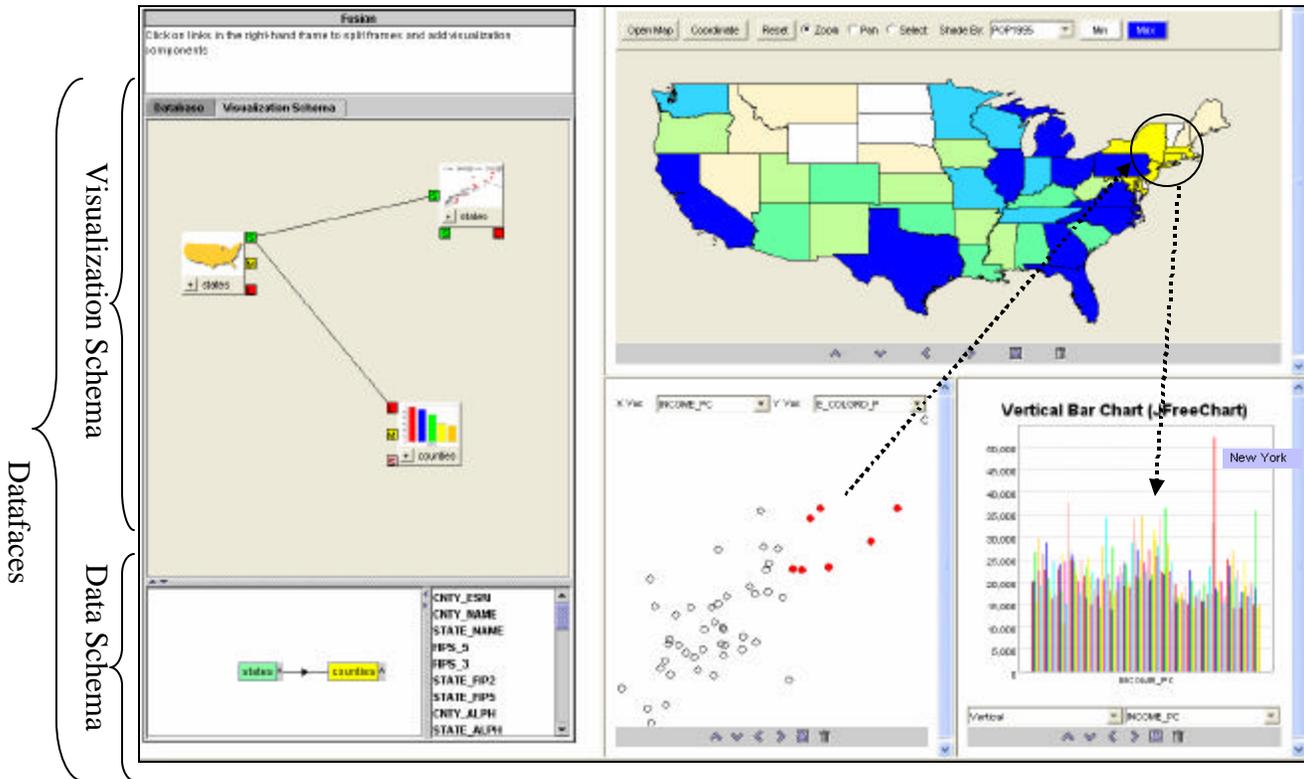
She then splits the bottom frame vertically and loads a *bar chart* and selects “Counties” relation to be loaded in the chart. She creates coordination between the map and the bar chart by a “select to load” association which implies that the data loaded in *bar chart* is decided by the data selected in the map. This allows the analyst to view details for counties belonging to states selected in the map. Selection in the *scatterplot* is also related to selection in map and hence a

selection in the *scatterplot* will result in selection in map which in turn will result in counties for selected states (in map) being displayed in the *bar chart*. She then configures the bar chart so that the height of bars is decided by the per capita income. She immediately observes an outlier (New York) in data corresponding to states with high percentage college graduates and high per capita income (Figure 15).



**Figure 14: Selection of states with high values of per capita income and percentage college graduates selects these states in the map showing that these states are located in the northeastern region.**

## Visualization Workspace



**Figure 15: Fusion coordinated visualization with tightly coupled actions between a Geographical map, Bar chart, and Scatterplot. Visualization workspace and Datafaces consisting of visualization schema and data schema is also shown.**

In this scenario we demonstrated the power of Fusion and Visualization Schema in constructing multi-view visualizations. The visualization was constructed quickly and was powerful in finding critical relationships. User had the control over data, views, and coordinations. An overview of the coordination structure was always visible and no programming was required to coordinate views dynamically and flexibly. Users can add, delete, or change view, data, or coordination at any time during the analysis. This flexibility allows rapid, effective, and customized construction of powerful multi-view visualizations.

### 1.7 Content

Chapter 2 reviews related literature, and provides a framework of the space that Visualization Schema fits within. As shown in Figure 16, the chapters are organized as per the columns i.e. the model, user interface, and Architecture. Each chapter explains contributions in terms of the

horizontal layers i.e. data schema, visualization schema, and coordinated visualizations. Chapter 3 describes the theoretical model of visualization Schemas. Chapter 4 describes the Visualization Schema user interface for coordination construction. Chapter 5 describes the Visualization Schema architecture that enables independent visualization tools to be coordinated. Chapter 6 concludes with limitations, contributions, and future work. Appendix A demonstrates Fusion and Visualization Schema with several additional scenarios.

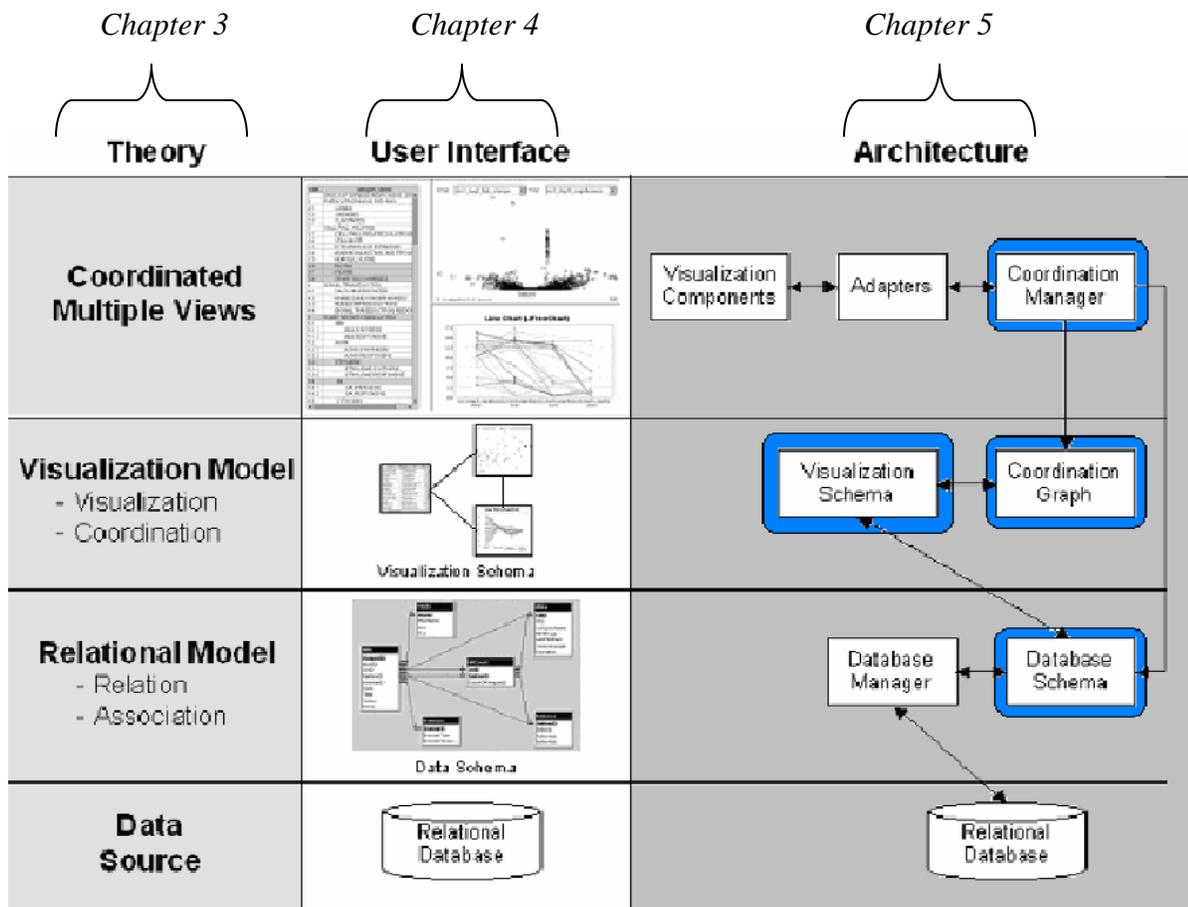


Figure 16: *Fusion* system diagram showing various layers of the system. Horizontal rows show the system layers whereas vertical columns show the model, user interface and architecture for each horizontal layer.

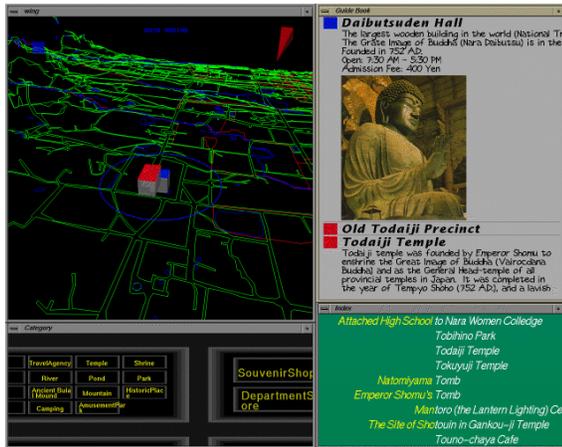
## Chapter 2 Related Work

### 2.1 Multiple View Visualization

Multi-view visualization shows two or more views to represent a single conceptual entity [5]. Multiple views are useful for exploring different aspects of related data [8]. Views can differ in their data or in visual representation of the data. Research on constructing visualizations has focused primarily on visualizing single relations or tabular data. Multi-view visualization tools such as DEVise [22], Spotfire [3], APT [23], Sage and SageBrush[36] allow mapping of data attributes to visual properties [Figure 17]. While Sage and APT use an automated approach under the “generate and test paradigm”, DEVise and Spotfire utilize form-based dialog boxes [30]. These tools have enabled users to construct useful multi-view visualizations of simple tabular data.

Visage describes the information-centric user interface paradigm[10]. In Visage, data objects are represented as first class interface objects that can be manipulated using basic operations, such as drill-down and roll-up, drag and drop, copy, and dynamic scaling. HomeFinder utilizes tight coupling between windows to support dynamic query filters and Details on Demand [4]. While the view coordinations are hard coded in *Spotfire*, it supports dynamic queries on simultaneously active multiple visualizations. VQE [24] enables queries and visualizations to be dynamically linked, i.e., queries can be created from visualizations and vice versa through direct manipulation (Figure 18).

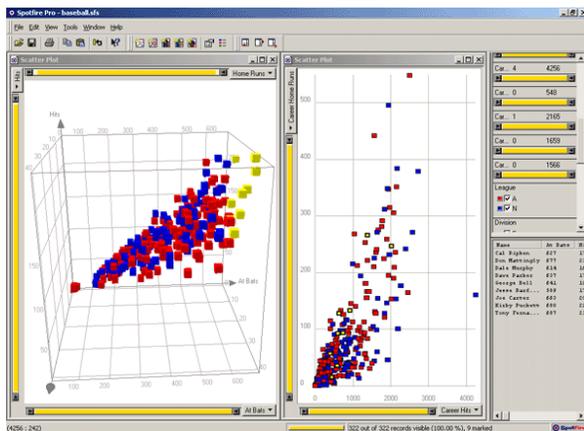
Wing [44]



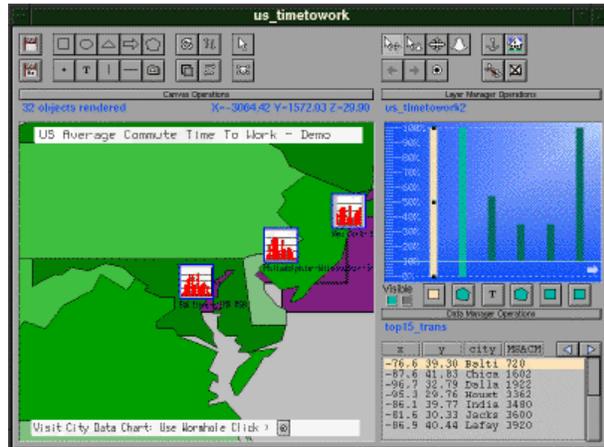
Home Finder



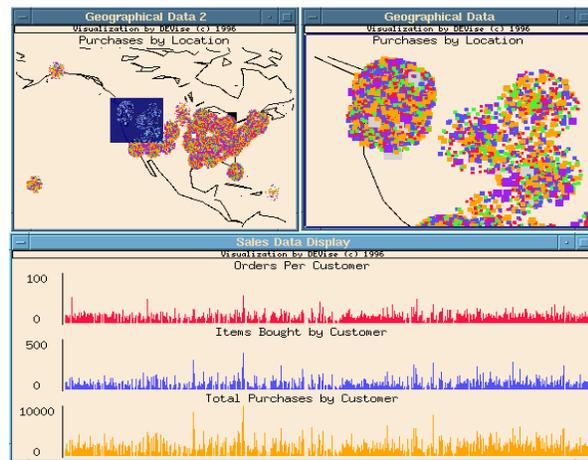
Spotfire



DataSplash



DEVis



Visage

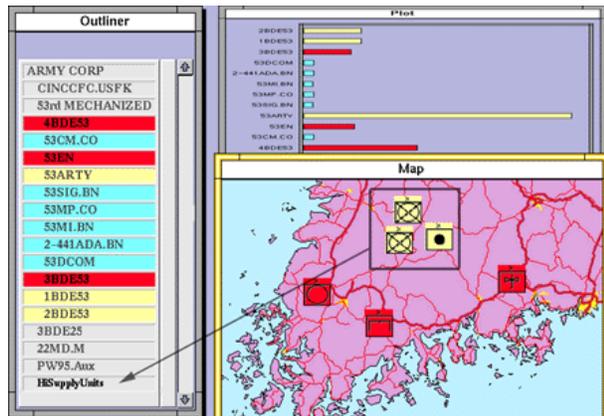
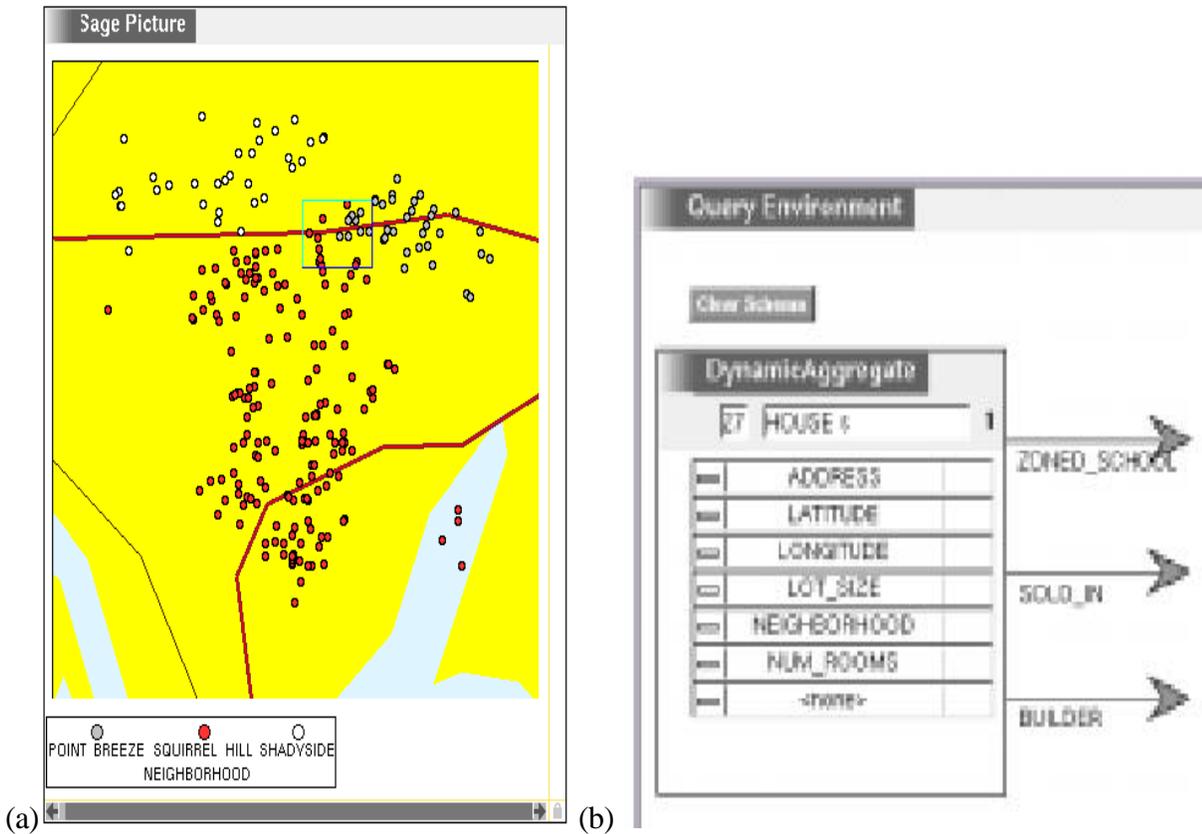


Figure 17 – Wing [44], DataSplash [4], Home finder[2], DEVis, Spotfire, and Visage are examples of how multiple-view visualizations support many problem domains.



**Figure 18: VQE (a) indicating the entire set of houses, color-coded neighborhood and (b) indicates selected houses have been dragged to VQE, where they become a dynamic aggregate. The arrows to the right constitute a menu for parallel navigation.**

Diamond is a visual data analysis package and a programmable environment that allows users to explore and manipulate tabular spreadsheet-like data (Figure 19). Its strength lies in its considerable variety of complementary graphical views available, and its dynamic interlinking of color [34].

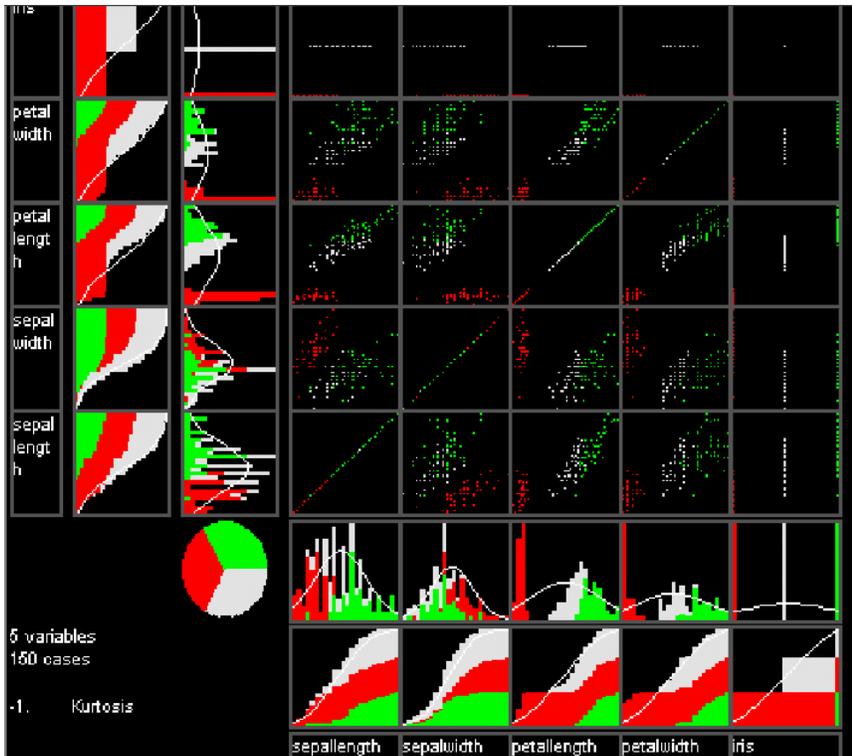


Figure 19: Diamond, a prototype system for interactive exploration of multidimensional data [35].

## 2.2 User constructed Visualization

The dataflow model provides an effective approach for constructing and coordinating visualizations [9]. Typically data flows through a network and undergoes processing at nodes. AVS [42] uses a user constructed schema for the specification of directed data flow and the processing network. AVS uses data-type specific input and output ports for coupling of modules.

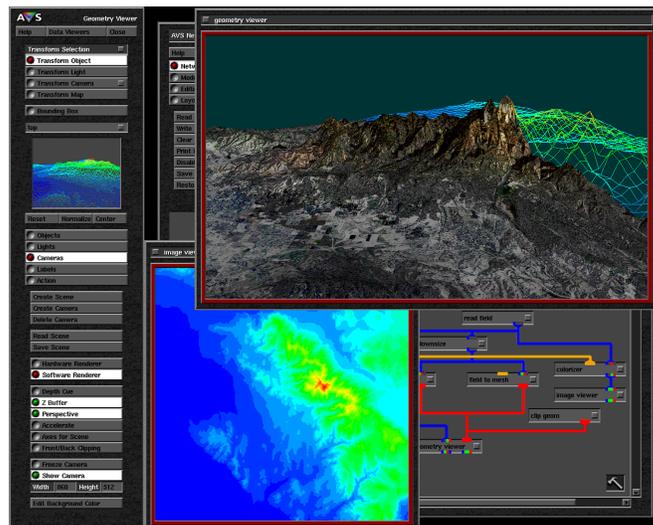


Figure 20 - AVS is a multiple-view visualization system that utilizes the dataflow model.



GeoVista allows users to construct a visualization design in a design window separate from the GUI window. It uses connectors to specify the coordination between components [41]. Dataflow systems such as AVS and GeoVista (Figure 23) enable flexible specification of data processing pipelines. Because of the focus on computationally intensive scientific-visualization applications, dataflow diagrams have evolved more as a representation for data processing rather than a specification for interactive visualizations or user interfaces [29].

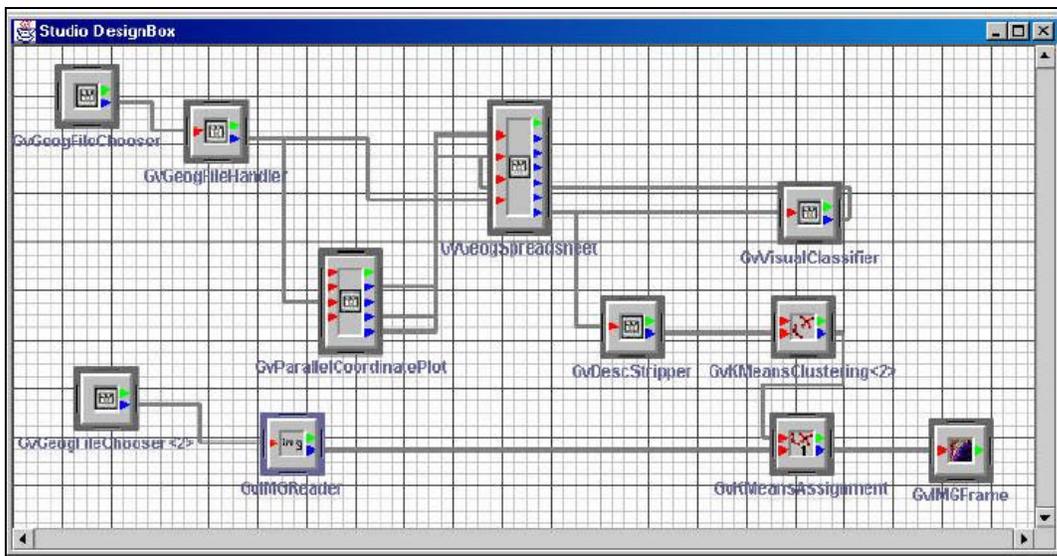


Figure 23: Geovista Design window allows users to create multi-view visualization by defining connections between components. A coordination is drawn between color coded connectors.

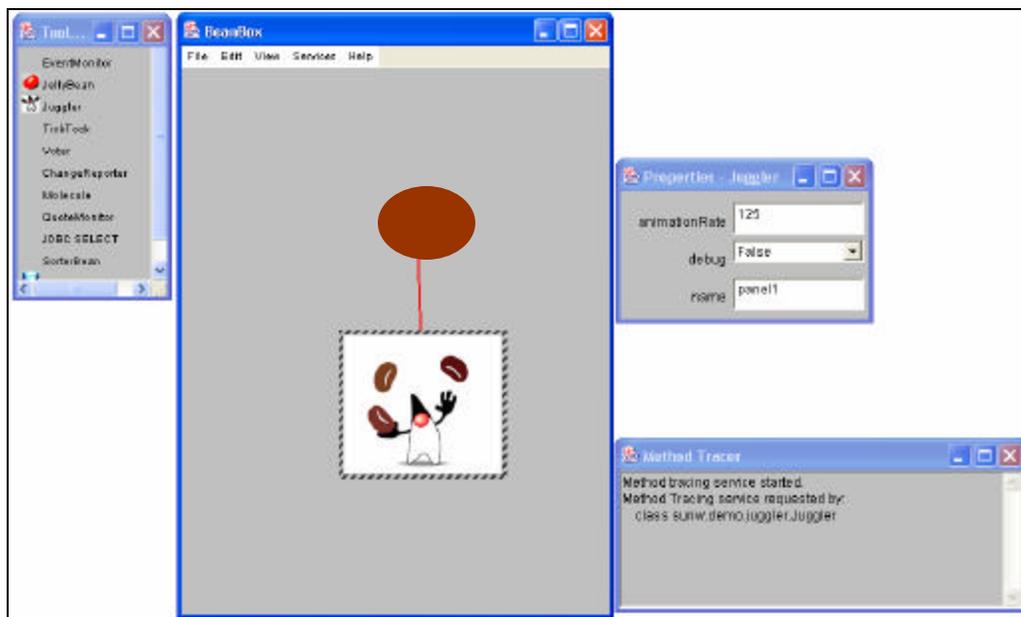


Figure 24 : Java’s BDK. Bean events can be tied to each other to build a component based application. BDK adopts a form based approach for specification of events for individual components.

ISYS is a dynamic, flexible platform for the integration of bioinformatics software tools and databases[1]. ISYS discovers, using *DynamicDiscovery*, tools and services that are relevant to the data.

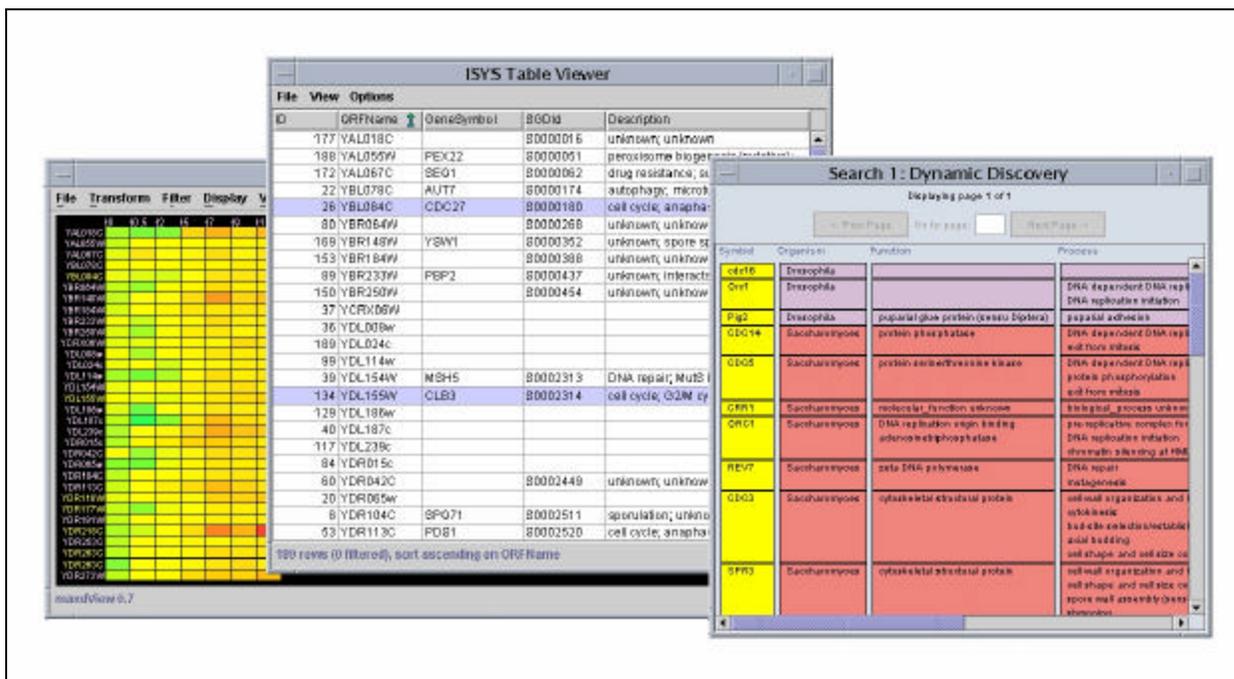


Figure 25: ISYS [25]. Showing multiple related views specialized for bioinformatics data. ISYS provides dynamic, flexible platform for the integration of bioinformatics software tools.

### 2.2.1 Snap-Together Visualization

Snap allows users to construct custom multi-view visualizations by defining coordination. Snap model is analogous to the relational model of underlying data [27].

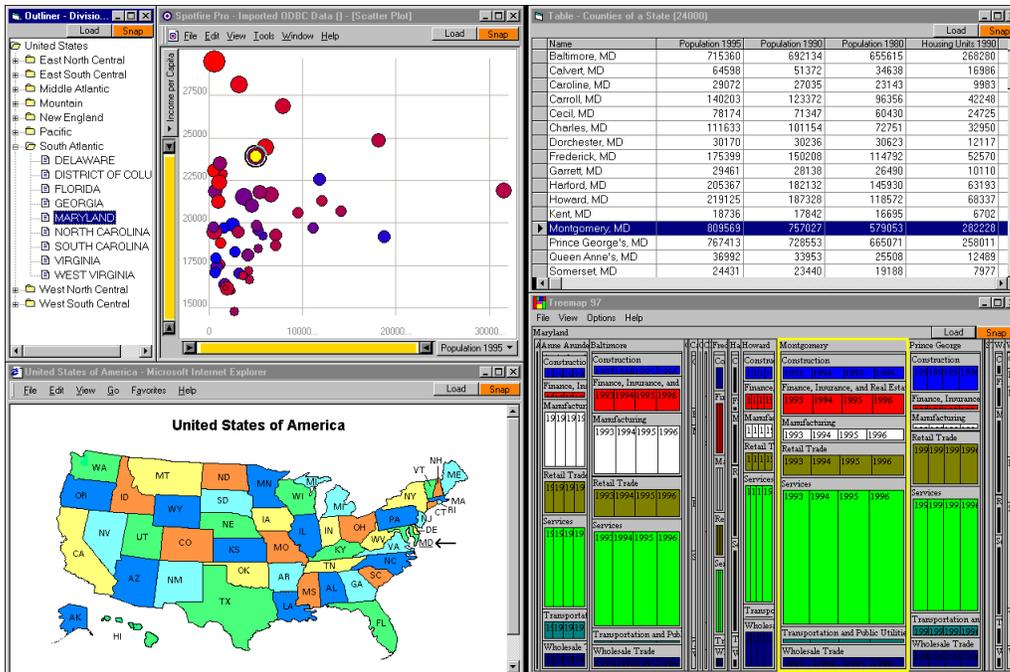


Figure 26 - Snap-Together Visualization allows coordination based on relational joins.

North presents a language and taxonomy for defining multiple view visualizations and their coupling [26]. The taxonomy includes several tasks and each task may translate to one or more user interface actions. Following visualization coordinations are included in the taxonomy

- Select ? Select (Brushing and Linking)
- Select ? Navigate (Overview + Details, Drill-Down, or Details on Demand)
- Navigate ? Navigate (Synchronized Navigation)

[5] presents guidelines for the design of multiple view systems. These guidelines describe when and how to use multiple views. The rules recognize the impact on user learning time, user memory, and machine computation [9].

Further research efforts on Snap have resulted in a form based UI (Figure 27 and Figure 28) for coordination specification and a web based architecture [9].

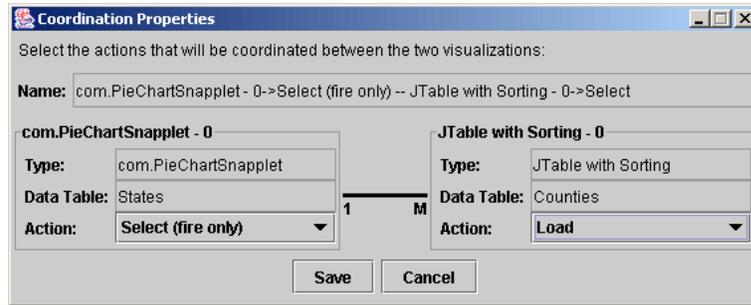


Figure 27 - The Coordination Properties window allows for the specification of a coordination.

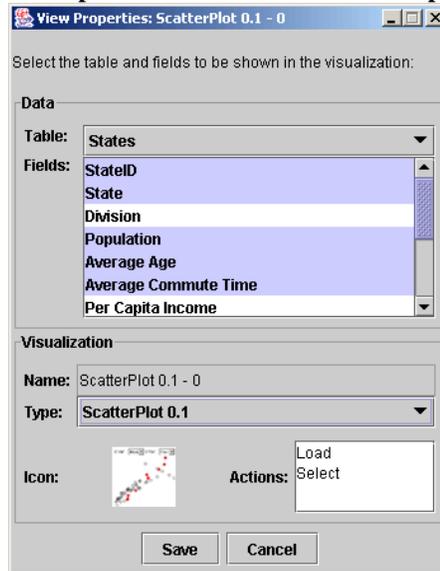


Figure 28 – The View Properties window allows for the configuration of the data and the visualization.

### 2.3 Relational Database Schemas

For exploring databases containing multiple relations, [12] and [19] enabled users to navigate through a database according to its database schema. These systems typically display only one tuple at a time. Users could follow relationships to related tuples in other relations.

VOODOO [11], Tioga-2, and SeeData [18] are all tools used for visualizing and constructing database schemas. In *VOODOO*, query takes the form of a tree that reflects the database schema, where every class or type reference in the schema is expanded [Figure 29].

In *SeeData* tool, the relations present in the database are drawn in a rectangular layout with each bar representing a single relation. The length of each bar is tied to a numeric statistic for a relation, e.g., number of attributes, while the color of bar represents categorical statistics

associated with relation, e.g., access method [Figure 30]. For extremely complex databases SeeData and OPPOSSUM [15] help users visualize and manipulate data schema themselves.

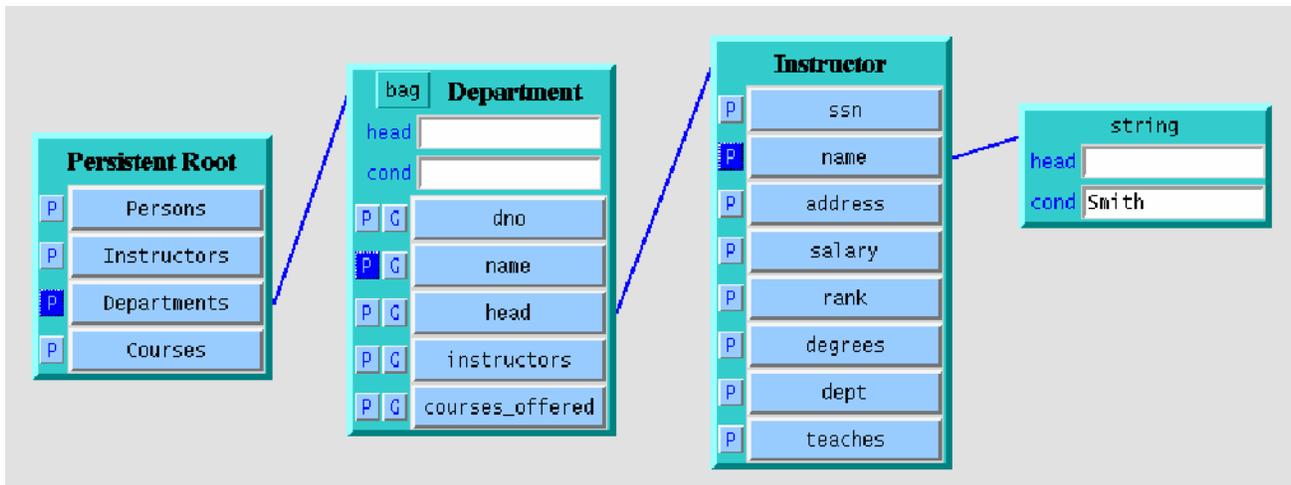


Figure 29: Diagram showing the visual representation of the query in VOODOO

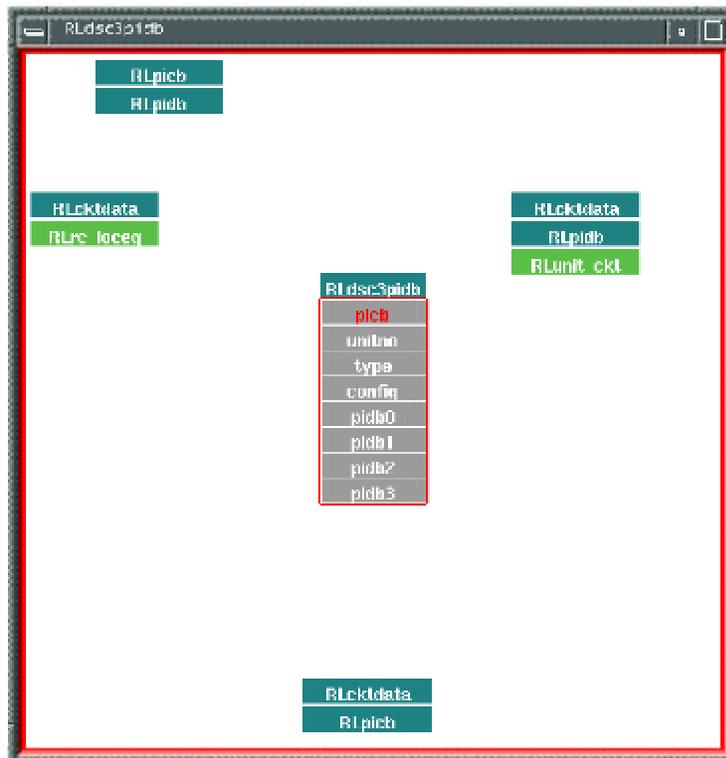
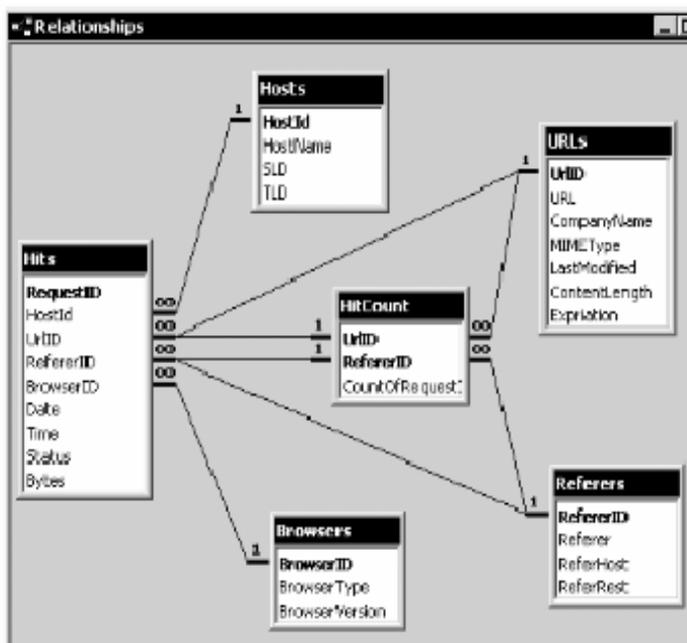


Figure 30: Associations View in Seedata. It shows the attributes in a current relation and the associated relations given in database specifications. Associations are directed. The position of the other relations in the window indicates their association with the current relation.

In *Tioga-2* user defined procedures are represented as a box and the output of one box, if compatible, can be connected to the input of another box. It uses the concept called “*Wormholes*” which is a viewer onto another canvas and hence associates objects in one visualization space directly with objects in a different visualization space.

In *Microsoft SQL Server and Microsoft Access* (Figure 31) a relation and its cardinality is clearly visible. Also the primary keys in the table are highlighted. A Database Schema is visually represented as a graph, with relations as nodes and relationships as links.

RMM uses a hypertext schema approach similar to visualization schemas [16]. Tuples match to web pages, relations map to index pages, and relationships map to hyperlinks. RMM generates hyperlinks based on join relationships. Key elements of RMM’s data modem (RMDM) are shown in Figure 32. An application's design is specified with an RMD diagram (Figure 33), constructed from RMDM's elements.



**Figure 31: Diagrammatic data schema for a database of website hits in Microsoft Access.**

<i>E-R Domain Primitives</i>	Entities	
	Attributes	
	One-One Associative Relationship	
	One-Many Associative Relationship	
<i>RMD Domain Primitives</i>	Slices	
<i>Access Primitives</i>	Uni-Directional Link	
	Bi-Directional Link	
	Grouping	
	Conditional Index	
	Conditional Guided Tour	
	Conditional Indexed Guided tour	

Figure 32: The elements of the RMM Data Model

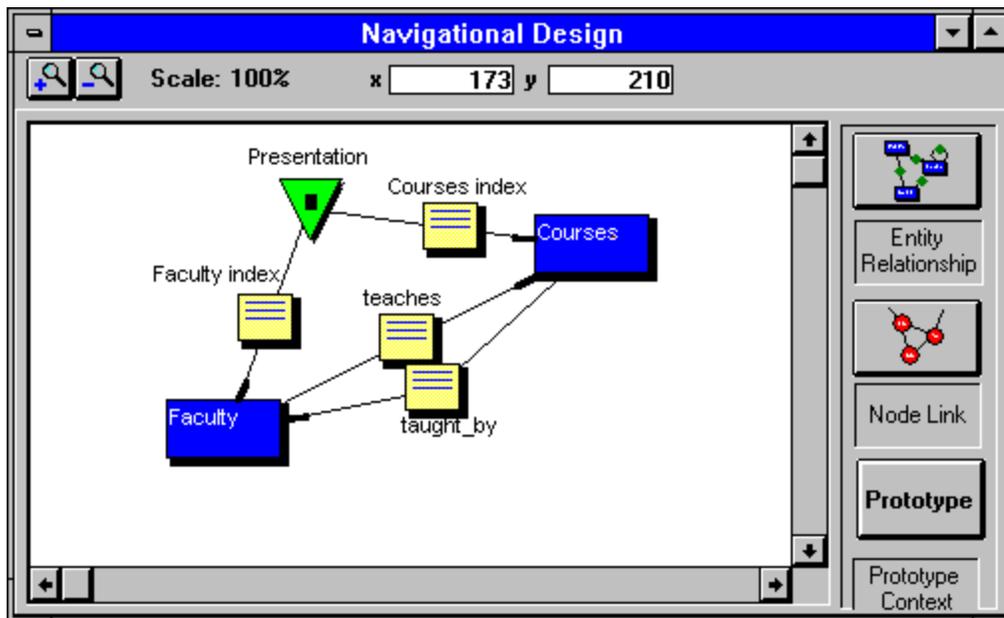


Figure 33: RMD diagram for *Faculty-Course* website Navigation Design.

## Chapter 3 Visualization Schema: Model

### 3.1 Motivation

An enhanced visualization model was needed to support flexibility in visualization design, decoupling of visualization components, providing support of a wider range of actions, custom operations on data (e.g., data mining, custom column names, custom data format, etc.) before it is loaded into visualizations, multiple data sources, and multiple relationships.

The initial Snap model involved parameterized queries and primary key and foreign key actions. Although the model was powerful enough to allow coupling of visualizations, it lacked flexibility and was not easy to configure. The designer would have to specify, by constructing parameterized queries, the relationship between components to correctly couple them. The component would need to distinguish between primary key and foreign key actions. Support for multiple databases, multiple relationships, and multiple keys was lacking. Visualizations could only be coupled on the basis of relationships that existed in the database and construction of these relationships was not facilitated by Snap. The user had to rely solely on existing relationships between database tables which were constructed at the time of database creation. Frequently in data exploration and analysis there is a need to analyze data which might be unrelated at the database level. This is required to explore interesting trends and to unveil possible relationships. Visualization schema model should address these issues and simplify the requirements for swift construction of multi-view visualizations.

The conceptual model for Snap was enhanced to allow for the much needed flexibility in construction of multi-view visualizations [30]. The components are now responsible for firing and receiving events based on primary keys of the relation they encapsulate. The model no longer requires foreign key actions.

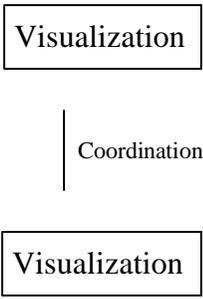
This research proposes further enhancements to the Snap model to present the Visualization Schema model. These enhancements include generalization of the concept of coordination as any joinpath in the database. A joinpath is a sequence of <table, key> values defining a database

relationship. Visualization Schema model allows for coordination between visualizations encapsulating relations from multiple separate databases. Users can coordinate relations from different local or remote databases. Construction of *Virtual relationships*, i.e., relationships which do not exist in the databases but are needed for the purpose of effective exploration and analysis of data, is also supported. More than one relationship might exist between two relations in the database schema and hence users can specify a relationship to be used for coordination between two visualization components.

### **3.2 Visualization Schema Theory**

The conceptual model for Visualization Schema is based on the relational database model. The relational database model is based on solid theoretical foundations and this has helped in development of techniques for effective design of data schemas and relationships. We hope to facilitate construction of effective multi-view visualization by using visualization schemas. Visualization schemas attempt to extend the relational data model to the visualization domain by establishing a direct correspondence between visualization concepts and relational data concepts [Table 1].

**Table 1: Schema Primitives [29]. Comparison of relational database and Fusion visualization system**

	<b>Relational Databases</b>	<b>Fusion Visualization</b>
<b>Design goal</b>	Data design	Visualization design
<b>Design method</b>	Data schema	Visualization schema
<b>Designer</b>	Data owner	Data owner, developer, presenter
<b>Adaptability</b>	Flexible	Flexible
<b>Rate of change</b>	Rapid, dynamic	Rapid, dynamic
<i>Layers:</i>		
<b>Theory</b>	Rel. data model	Visualization Schema model
<b>User interface</b>	Rel. data schema	Visualization Schema
<b>Architecture</b>	Rel. DBMS	Fusion visualization server
<i>Schema Primitives:</i>		
<b>Conceptual</b>	Relation	Vis. component
	Tuple	Visual item
	Attribute	Visual property
	Join association	Coordination
<b>Diagrammatic</b>		

- *Visualization component* ? data relation. A visualization component is a view that displays a data relation or query result (we assume that queries or database “views” are integrated into the data schema like relations). A visualization component can implement a specific visualization type (e.g., geographical map in Figure 15).

- *Visual item* ? data tuple. Data tuples are displayed as visual items in a visualization component. For example, a tuple is displayed as a dot in the *scatterplot* in Figure 15.
- *Visual property* ? data attribute. Data attributes are used by visualization components to compute graphics. Users map data attributes to component-specific visual properties. For example, a data attribute is mapped to the height of a bars in a bar chart, causing tuples to be visually arranged according to their value for that attribute (Figure 15).
- *User interaction* ? tuple subset selection. A user interaction in a visualization component selects a subset of tuples from the displayed relation, analogous to performing a selection query, and typically alters the visual display of those tuples. For example, a user highlights a set of outlier tuples in the bar chart by directly selecting them (Figure 15).
- *Visualization coordination* ? data join. Coordination links an interaction in one component to an interaction in another component, by equating the corresponding subset selections according to a join between the components' relations. For example, brushing-and-linking between a map and the scaperplot encapsulating the same relation is a coordination of the “select” actions across the implicit one-to-one join association (Figure 15).
- *Domain of Visualization Attribute* ? *Domain of relation attribute*: Domain of a visualization property is defined as a set of permitted values for that property. The domain of a property  $A_i$  is represented as  $\text{dom}(A_i)$ . For example, a domain of actions supported by a visualization would include *select*, *highlight*, *load*, etc.

- 

The Snap model formalizes multiple view visualizations and enables the establishment of a specification language [26]. We extend this specification and introduce the concepts of visualization relationship, Coordination, joinpath, and coordinated pair.

- *Visualization relationship* for attributes A and B of visA and visB respectively is defined as a subset of the cartesian product of domains of visual attributes A and B .i.e. subset of  $\text{dom}(A) \times \text{dom}(B)$ . A Visualization relationship between visA and visB is described as

$$((\text{visA}, \text{attributeA}), (\text{visB}, \text{attributeB}))$$

For example, in Figure 15 a “select to load” visualization relationship exists between the map component and the bar chart component. This relationship is expressed as

$$((\text{map}, \text{select}), (\text{bar chart}, \text{load}))$$

A visualization relationship is *commutative*, i.e.,

*IF ((visA, attributeA),(visB,attributeB))  
Then ((visB, attributeB),(visA,attributeA))*

A visualization relationship is also *transitive*

*IF ((visA, attributeA),(visB,attributeB))  
and ((visB, attributeB),(visC,attributeC))  
then ((visA, attributeA),(visC,attributeC))*

This means that there is no difference in the model and in event propagation between direct and indirect actions. A *direct* action is a user action performed by the user on a visualization component whereas an *indirect* action is an action which occurs as a result of coordination between components. An indirect action on a view should have the same effect as the direct action. For example, selection of a row as a result of “select to select” coordination with another view would have the same effect as if the row was directly selected by the user.

- *Coordination* between visA and visB is defined by a visualization relationship, a database relationship, and itemsA and itemsB belonging to visA and visB. We will use the term *joinpath* to indicate database relationship.
- *Joinpath*: A joinpath between relations A and B is defined by A, B and a sequence of <relation, key > pairs

[(relationA, keyA), (relationX, keyX),....., (relationB, keyB)]

The number of pairs in the joinpath is referred to as the *length* of the joinpath. A joinpath can span across multiple databases and hence enables specification of coordinations spanning multiple databases.

- The specification for a *coordinated pair* of visualization is

[(visA, actionA, itemA),joinpath,( visB, actionB, itemB)]

For example, in Figure 15 , the map and table view for a coordinated pair defined as

[(map, select, items-map),joinpath,( bar chart, load, items-bar chart)]

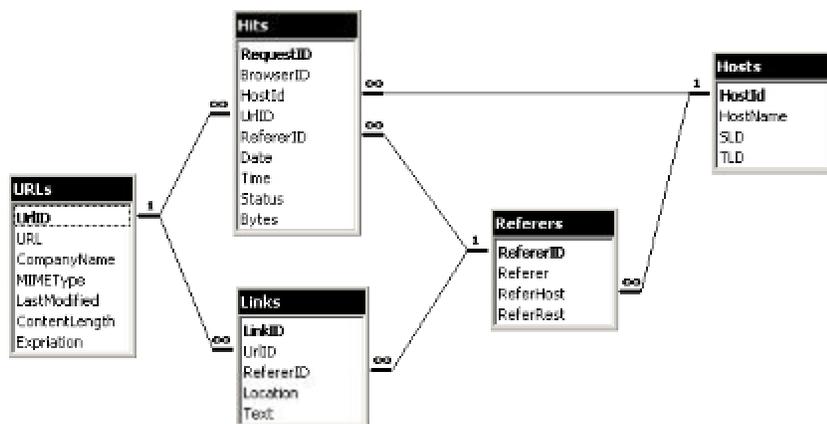
A coordination is commutative (bidirectional)

*If [(visA,actionA,itemA),joinpath,( visB,actionB,itemB)] then  
[(visB,actionB,itemB),joinpath,( visA,actionA,itemA)]*

### 3.3 Coordinations and Joins

An important advancement in this model is the generalization of coordinations and interactions [29]. A coordination is generalized to any single or compound join association, including one-to-one, one-to-many, and many-to-many associations. Join associations for coordinations are automatically derived and executed from data schemas, eliminating the need for user-defined parameterized queries for joins. Furthermore, interactions are generalized to tuple subset selections, enabling users to act on single or multiple tuples. Chained coordinations cascade across arbitrary associations, beyond the previously limited one-to-one cascading [27].

Four cases demonstrate how generalized coordinations correspond to various joins in the data schema. These are demonstrated using the example data schema and multiple-view visualization for website hits shown in (Figure 34) and (Figure 35)



**Figure 34 – An example data schema for a database of hits to our website. “URLs” stores information about pages on our website. “Referrers” stores information about external websites that have links to our website. “Hits” stores information about each hit, including a reference to the page requested in “URLs” and the external referring site in “Referrers”.**



Figure 35 – An example multiple-view visualization constructed with Snap for the database shown in Figure 34. The website map generated from the URLs is shown in the TreeView (top left). Selecting a page in the map displays the page in the web browser (top right), and displays the distribution of hits to that page in the scatter plot (top center). Selecting pages also highlights referring sites listed in the table view (bottom left).

Likewise, selecting referring sites highlights pages linked to, and shows their hits in the plot. Clicking a referrer shows its page in the other web browser (bottom right).

- *Self join:* A coordination can be established between two visualization components that display the same relation. In this case, the coordination corresponds to the implicit one-to-one join association that exists between the relation and itself.

For example, the TreeView visualization component displays the URLs relation, using the URL page pathname attribute to display the tuples as a website tree structure. A web browser component also displays the URLs relation, using the URL attribute to display actual pages represented by the tuples. The “select” action of the TreeView is coordinated to the “navigate” action of the web browser across the self join. When users select a tuple in the TreeView, there is no need to actually perform a join since it is a self join, so the same tuple is used to navigate the web browser to the selected page.

TreeView ← URLs → Web browser

- *Single join*: A coordination can be established between two components whose relations have a direct join association in the data schema. The relational model provides two primitive types of direct join associations: one-to-one and one-to-many.

For example, in addition to the TreeView of the URLs relation, the scatter plot displays the Hits relation, showing all hits to the website by date and time. The “select” action of the TreeView is coordinated to the “load” action of the plot, using the direct join association between the URLs and Hits relations. The data schema indicates this is a one-to-many association. Selecting tuples in the TreeView joins those tuples to the Hits relation to find all the hits to the selected pages. The resulting Hits subset is then loaded and displayed in the plot, essentially filtering out hits to any other pages. This enables users to drill down from pages to hits.

TreeView ← URLs ↔ Hits → Scatter plot

- *Compound join*: A coordination can be established between two components whose relations have an indirect association via one or more intermediate relations in the data schema. This requires a compound join, concatenating each of the direct joins along the indirect association path. Compound joins enable more complex associations such as many-to-many. For example, in addition to the TreeView of the URLs relation, the TableView displays the Referrers relation, showing an alphabetical list of all the websites that link (refer) readers to the URLs website (alternatively, it might be interesting to show referrers geographically). The “select” action of the TreeView is coordinated to the “select” action of the TableView, using the compound join from URLs to Hits to Referrers. The concatenation of the one-to-many and many-to-one joins creates a many-to-many join. Selecting tuples in the TreeView joins those tuples to the Hits relation and then to the Referrers relation to identify the websites that actually sent readers to the selected pages, and then selects them in the TableView. Since coordinations are bidirectional, the reverse interaction also occurs. This example illustrates brushing-and-linking across a many-to-many association.

TreeView ← URLs ↔ Hits ↔ Referrers → TableView

- *Multiple alternative joins*: A coordination between two components whose relations have multiple alternative join associations connecting them requires the selection of one of the join associations for use in the coordination. In the compound join example above, an

alternative is the compound join through the Links relation. This alternative would have a different effect. Selecting pages in the TreeView would indicate all the websites in the TableView that have links to those pages, rather than the websites that actually referred readers and generated hits.

TreeView ← URLs ↔ Hits ↔ Referrers → TableView

TreeView ← URLs ↔ Links ↔ Referrers → TableView

### **3.4 Relationship between Visualization relationship and Database relationship**

Visualization relationships exploit existing database relationships for propagating events between coordinated visualizations. Standard Visualization techniques like brushing and linking, Overview and detail, drill down, etc., can all be expressed in terms of visualization relationships. Each visualization action can correspond to either one or many tuples. Mouseover, for example, corresponds to only one tuple per event whereas select can correspond to many tuples. Depending on the relationship between visualizations' relations a visualization can get one or many tuples as a result of event propagation. It is up to the visualization to decide how it handles conflicting cardinalities between database relations and visualization actions. E.g. it is up to the visualization to decide what to do when it receives multiple tuples for a single cardinality action (mouseover or scroll).

Depending on the database relationship we can have the following cases:

- One to One :

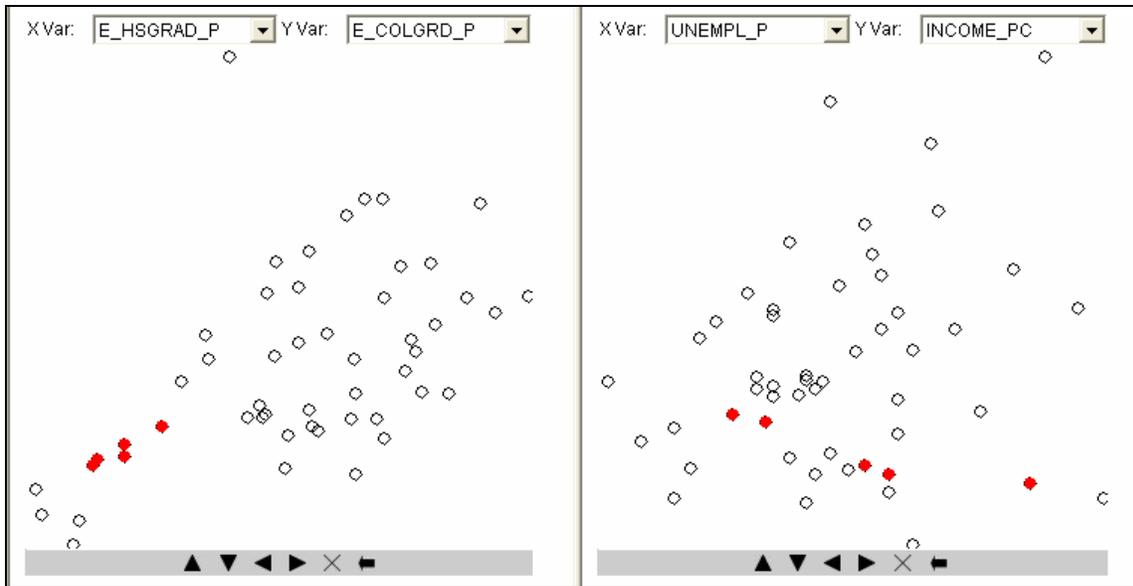
This can occur in case of any kind of join i.e. self join, single join, or compound join.

Some of the examples using this relationship are:

- *Brushing and Linking:* An exploratory data analysis (EDA) technique used when displaying a set of data items in multiple visualizations [27]. When users select items in one visualization, those items are automatically highlighted in all the visualizations (Figure 36). A common example is brushing scatter plots [6].

In terms of visualization relationship this can be expressed as

((visualizationA, select), (visualizationB, select))

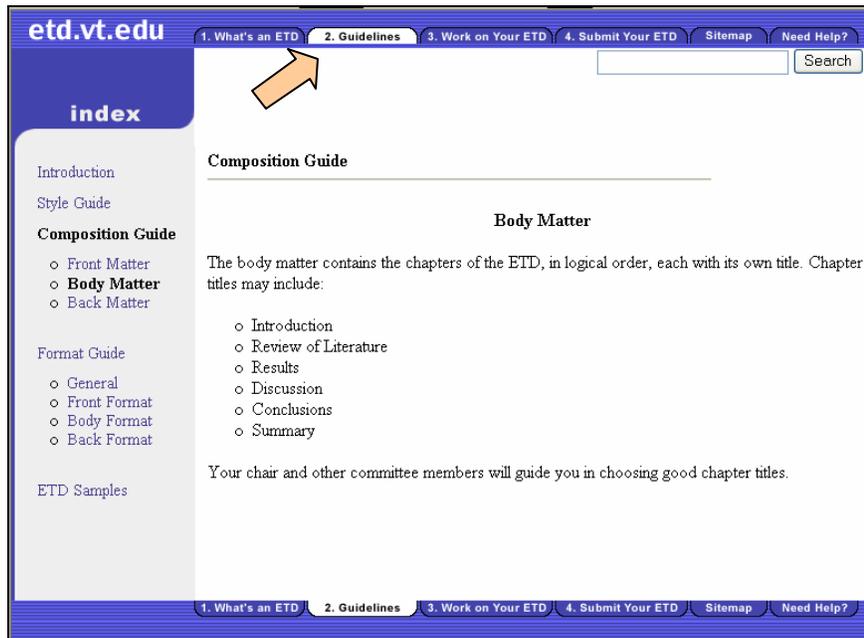


**Figure 36: Brushing and linking across Scatter plots encapsulating states data indicates that states having low percentage of college and high school grads (left) have low per capita income (right).**

- *Overview and Detail*: Selecting an item in the overview visualization navigates the detail visualization to the corresponding details (Figure 37). Items are represented visually smaller in the overview. This provides context and allows direct access to details [27].

In terms of visualization relationship this can be expressed as

((visualizationA, select), (visualizationB, scroll/zoom/...))



**Figure 37: Overview and Detail with web Frames.**

- Synchronized scrolling: Two views scroll together so that related data is always visible. Users can conveniently scroll through multiple corresponding data sets.

In terms of visualization relationship this can be expressed as

((visualizationA, scroll), (visualizationB, scroll))

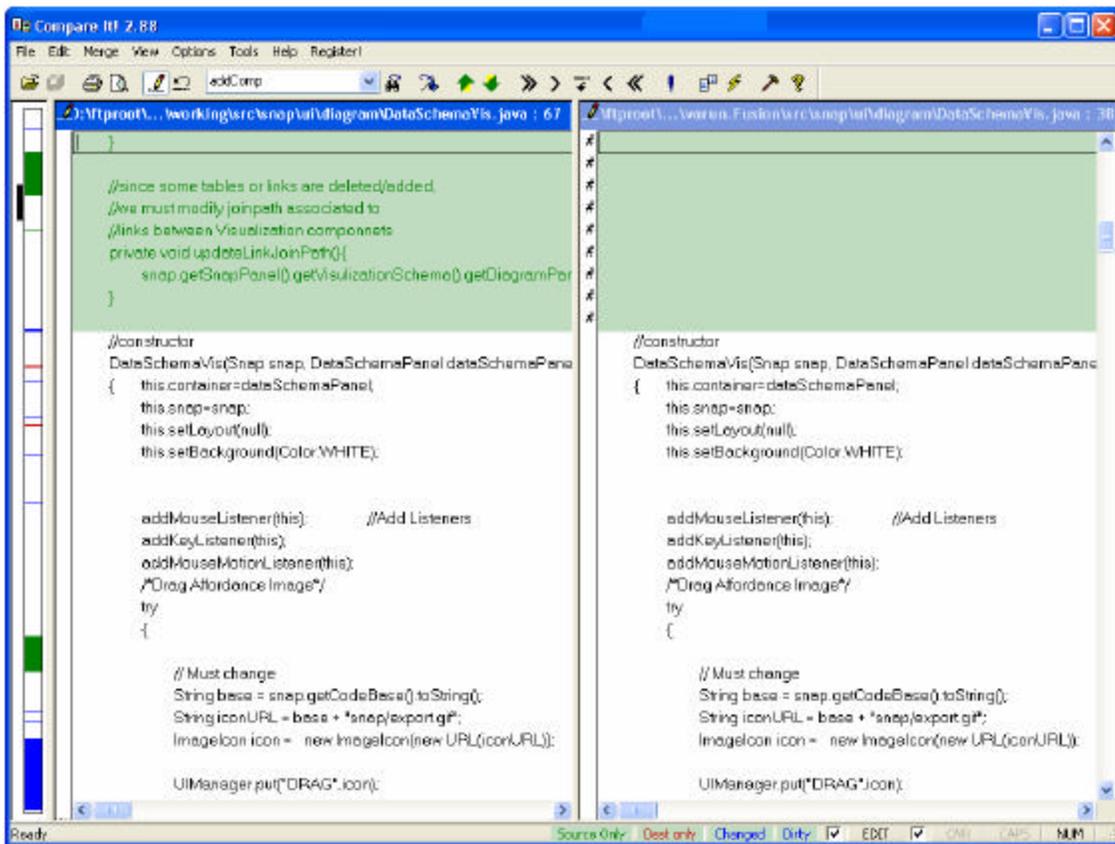


Figure 38: Synchronized scrolling in *Compare it!*, a file compare utility [14].

- One to Many:

This can occur only in case of single or compound join.

- *Drill Down*: Allows users to navigate down successive layers of a hierarchical data set. Selecting a parent item in one visualization loads children items into another visualization [27].

In terms of visualization relationship this can be expressed as

((visualizationA, select), (visualizationB, load))

- Many to Many:

This can occur only in case of single or compound join. This relationship generally occurs because of a combination of one-to-many relationships between relations.

- *Synchronized load*: Two or more views load the same data simultaneously as a result of the same direct user event. In terms of visualization relationship this can be expressed as

((visualizationA, load), (visualizationB, load))

## **3.5 Discussion**

The Visualization schema model generalizes the concept of coordinations between components as any joinpath in the data schema. This enables views encapsulating relations associated by a compound join to be coordinated. The Visualization schema model also presents a language for specification of coordinated multiple view visualizations. This provides a solid foundation for further research.

### **3.5.1 Future Work**

#### *3.5.1.1 Data Mining*

Data mining algorithms could be integrated in the data schema and visualization schema [28]. This will allow results of mining algorithms to be used to calculate associations at run-time such as statistical significance or inference rules. Mining algorithm components shall be selected and specified in the data schema as a part of a virtual relationship. In the visualization schema, mined associations shall be exploited for coordination. Users shall specify various algorithms and mining constraints.

#### *3.5.1.2 Normalization Rules*

Database Normalization rules enable standard and efficient database design. Similarly, Visualization normalization rules can result in efficient visualization design. These rules would act as guiding principles for design of complex multi-view visualizations. These rules could include specifications for maximum number of views, maximum number of coordinations per visualization, what actions to coordinate, etc.

### **3.5.2 Example Systems**

We describe some common multi-view systems using Visualization Schema model.

### 3.5.2.1 Windows Explorer

Windows explorer is a multi-view visualization for visualizing files and folders on a hard disk [Figure 39]. It can be described using the model as

((Tree-view, select),( Details-view, load))

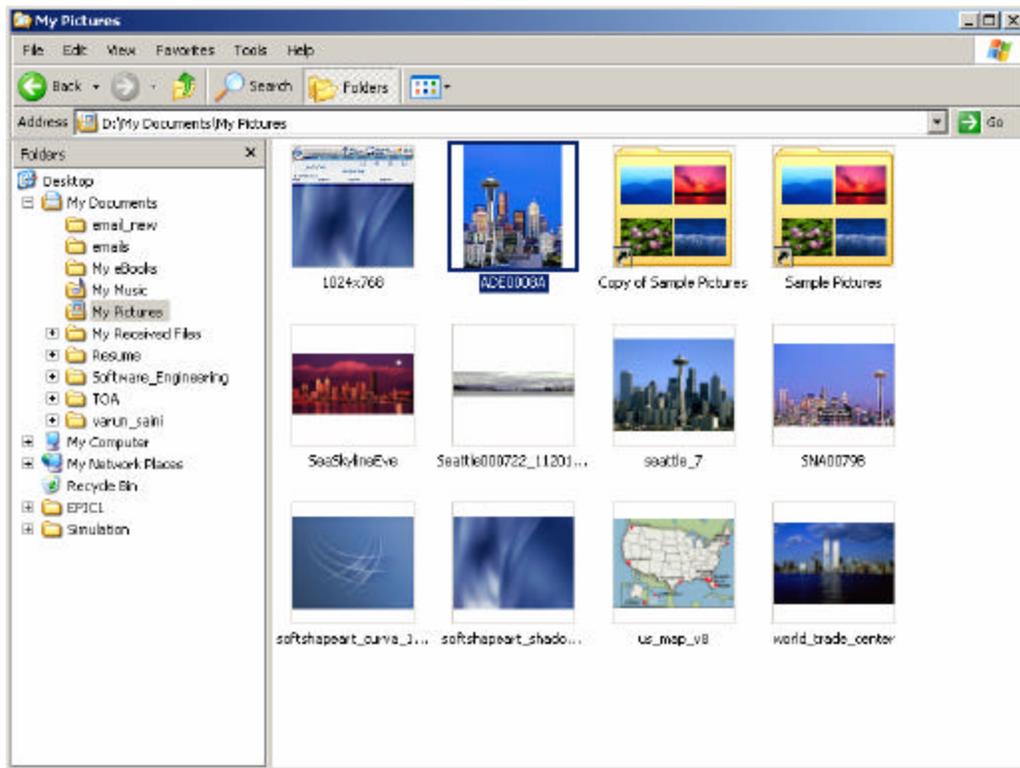


Figure 39: Microsoft Windows Explorer displaying two views. The tree-view showing folders on the left and the details-view on the right.

### 3.5.2.2 Javadocs

Javadocs are documentation files for Java program [Figure 40]. They are typically presented with three views. The left top frame shows the package view; the bottom left view shows the class view and the right frame shows the details-view for the classes. This can be expressed as

((package-view,select),(class-view,load))

((class-view,select),(details-view,load))

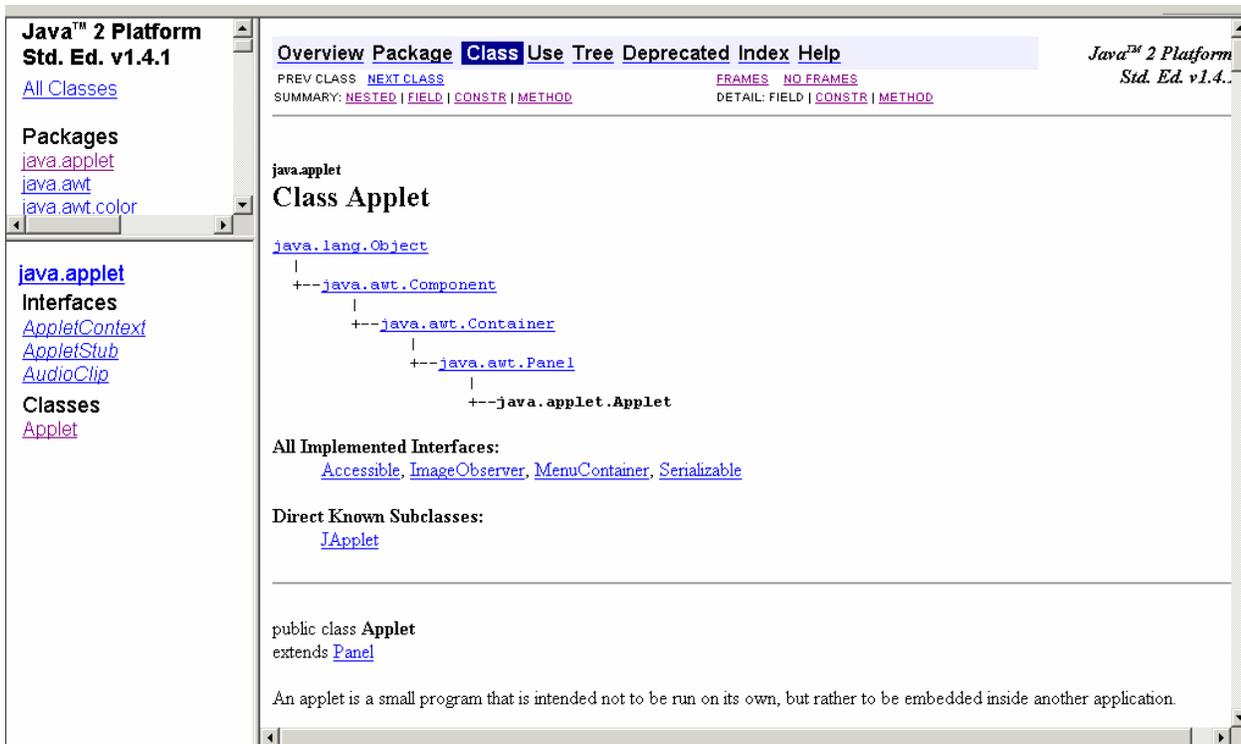


Figure 40: Javadocs for Java SDE 1.4.1. The left top frame shows the package view, the bottom left view shows the class view. The right frame shows the details view for the classes.

### 3.6 Summary

This chapter presented the visualization schema model and a language for specifying multi-view visualizations and coordinations. The Visualization schema model generalizes the concept of coordinations between components as any joinpath in the data schema. Inclusion of joinpath in coordination description enables specification of coordinations spanning multiple databases.

## Chapter 4 Visualization Schemas: User Interface

### 4.1 Overview

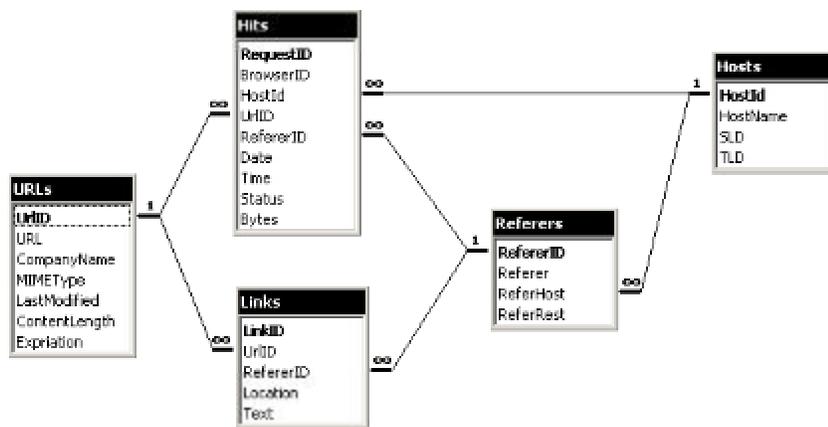
Visualization Schemas provides a user interface for constructing and coordinating visualizations dynamically and flexibly without programming. Visualization schemas makes the relationship between multiple views apparent through appropriate perceptual cues (“Rule of self-evidence” [5]). Visualization Schemas is built on top of database schemas (Figure 41). The primary purpose of visualization schema is to provide a level of flexibility in visualization design which is comparable to that of database design.

As in the database world and relational data schemas, it is important to provide the user with a visual, directly-manipulable means of constructing visualization schemas. Most popular database systems support visual construction and updating of database schemas. This allows users with basic knowledge of relational database concepts to create and manipulate database schemas with ease. We try to bring the same level of flexibility to the creation of visualizations schemas. By using visualization schema, the user shall be able to get an overview of existing visualization coordinations, create multi-view visualizations, understand and define relationships and coordinations, choose different views, change the data at run-time, change the relationships at run-time, save the schema and reload it, etc. Visualization schema helps the user in creating a visual and mental concept for the organization and interactions of a coordinated multi-view visualization.



graph represent instantiated visualization components. Edges represent coordinations between components.

As an example for the following description of visualization schemas, we construct a custom multiple-view visualization for the database of website hits whose data schema is shown in Figure 42. We construct the visualization shown in Figure 44 for exploring the website, time sensitivity of hits, and referring pages. The visualization schema for designing this visualization is shown in Figure 43.



**Figure 42: ( also Figure 34) An example data schema for a database of HTTP hits to our website. “URLs” stores information about pages on our website. “Referrers” stores information about external websites that have links to our website. “Hits” stores information about each hit, including a reference to the page requested in “URLs” and the external referring site in “Referrers”.**

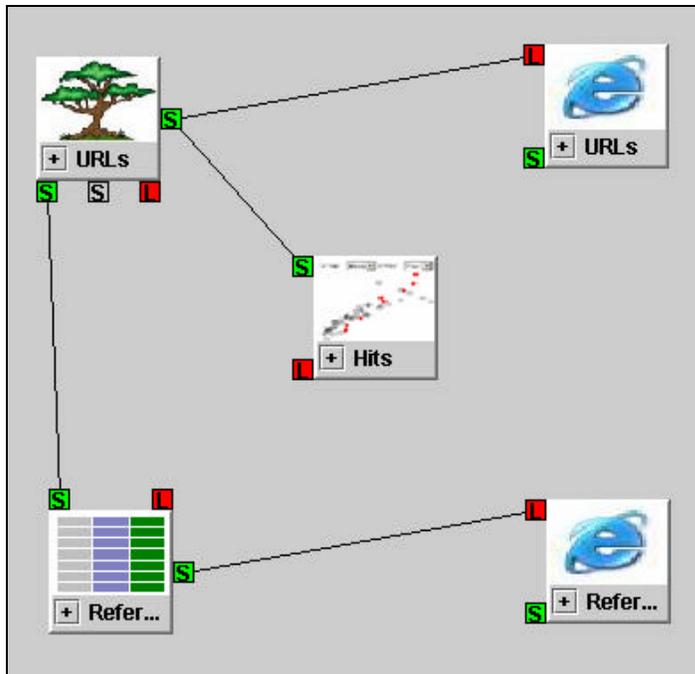


Figure 43: Visualization schema for the multiple-view visualization shown in Figure 44

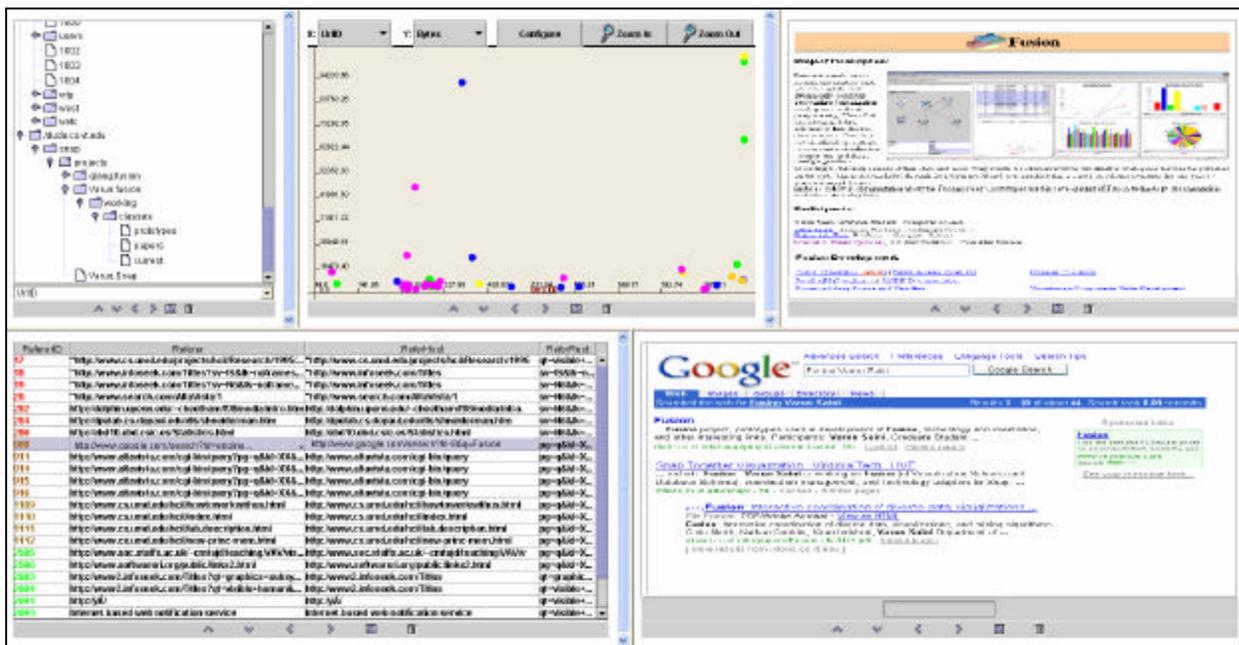
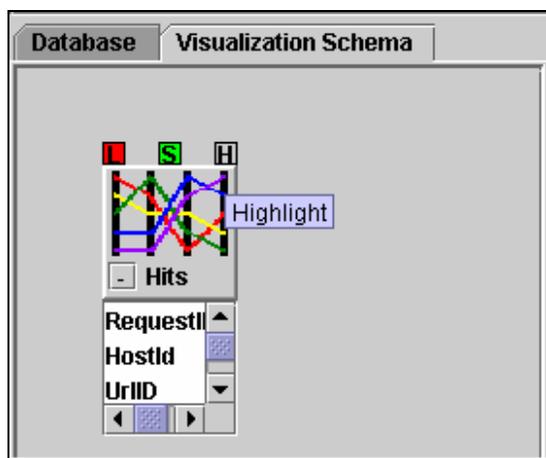


Figure 44: (also Figure 35) An example multiple-view visualization constructed with Snap for the database shown in Figure 42. The website map generated from the URLs is shown in the TreeView (top left). Selecting a page in the map displays the page in the web browser (top right), and displays the distribution of hits to that page in the scatter plot (top center). Selecting pages also highlights referring sites listed in the graphical-table view (bottom left). Likewise, selecting referring sites highlights pages linked to, and shows their hits in the plot. Clicking a referrer shows its page in the other web browser (bottom right).

## 4.2.1 Nodes

Nodes represent instantiated visualization in the visualization schema. Opening a new visualization component adds a new node to the graph. The actual instantiated visualization component that the node represents is displayed in a separate frame called the visualization workspace, tiled next to the visualization schema diagram. Users can minimize the workspace or the visualization schema to maximize the available space. A node is represented visually as a miniature icon of the component. It also displays the name of the data relation displayed in the component (Figure 45). Colored ports attached to the nodes represent the actions supported by the visualization component. Each port displays a descriptor of its corresponding action. Selecting the node in the workspace highlights the corresponding visualization in the workspace and thus helps in relating a node to its corresponding visualization. This becomes important as the user creates multiple instances of the same type of visualization (e.g., scatterplot).



**Figure 45: A node in the visualization schema represents an instantiated visualization component.**

Users can drag a relation from the data schema onto a component's node to display that relation in the component. Users can directly select attributes of the relation to include in the component. The list of data attributes loaded into a component can be seen by clicking on the "+" symbol at the bottom left of the node.

Users can drag nodes to rearrange the graph. While a node is being dragged all other nodes and links in the visualization are made semi-transparent. This allows the user to see the graph structure as they are modifying it.

To begin the example in Figure 44, a standard TreeView visualization component is selected from a menu. The TreeView node is displayed in the visualization schema (top left node). The URLs relation is loaded into the component by dragging the relation from the data schema browser. Snap prompts the user to specify the method used for generating a tree structure from the relation, and the user chooses the URL pathname attribute for this. In the visualization workspace, the tree structure of the website is displayed in the TreeView visualization component. Similarly, the URLs relation is loaded into a web browser component (top right node), choosing the URL pathname attribute for the browser to use for navigating to pages. The Hits relation is loaded and displayed in a scatter plot visualization component (top-center node). This creates the three views shown at the top of the workspace.

## 4.2.2 Edges

Edges between nodes in the visualization schema represent coordinations between visualizations. Edges indicate the actions that are coordinated at their endpoints and the joinpath (Figure 46). Dragging a link from a port on one node to a port on another node establishes coordination between the visualization components. Coordination can only be created between tables that have a joinpath in the database schema. Users can, in database schema UI, construct virtual relationships between tables, eventually resulting in a joinpath between the desired tables. Users decide the interface actions to coordinate by choosing the ports on each end of the edge. To keep the graph visually organized, ports will slide to the icon edge closest to the other coordinated icon and duplicate if needed (Figure 47). When the coordination is established, the actual instances of the visualization components will then immediately behave in the specified coordinated fashion. Coordinations can easily be altered by specifying a new joinpath (Figure 46) or different actions to coordinate. Visualization components and coordinations can be eliminated using right-click menus.

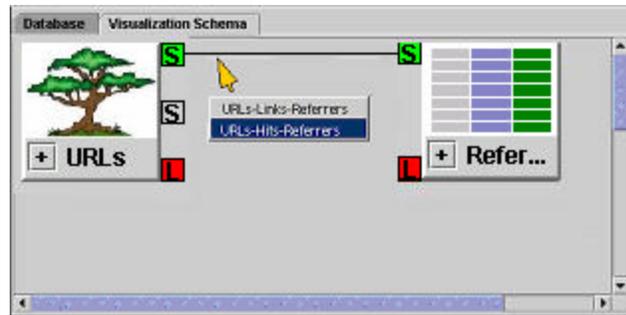


Figure 46: An edge in the visualization schema represents a coordination between visualization components. A coordination between the URLs and Referrers relations offers a list of two alternative many-to-many joins.

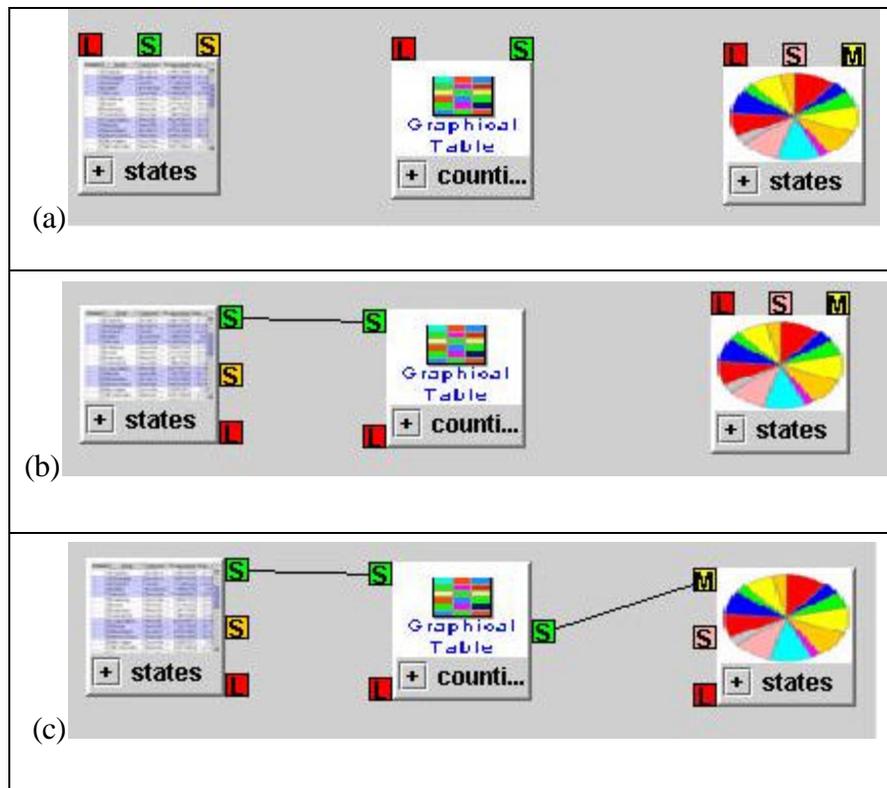


Figure 47: Port moving and splitting. (a) shows initial position of ports. (b) Ports move so that they are on the closest possible side. (c) Ports split to prevent the graph from becoming cluttered.

For each coordination, the system automatically determines the corresponding joinpath from the data schema. When coordinating, if there are multiple alternative joinpaths between the visualization components' relations, then the user can select the desired join path from a pop-up list (Figure 46). Fusion allows connection to multiple databases at the same time. Coordination between relations belonging to different databases can be established as long as a joinpath exists

between the two relations. Typically, user would build a joinpath by creating a virtual relationship between relations belonging to different databases.

Continuing with the Example in Figure 44, a link is dragged from the TreeView's "select" port to the web-browser component's "load" port in the visualization schema (top-most edge) as shown in Figure 43. This establishes a coordination between the components. The self join on the URLs relation is automatically chosen as the obvious join for the coordination. Now, selecting a page in the TreeView will load that URL in the web browser, causing it to display the actual web page. Similarly, dragging a link from the TreeView's "select" port to the scatter plot's "select" port establishes a brushing coordination between the TreeView and scatter plot. The one-to-many join between URLs and Hits is automatically chosen since it is the most obvious alternative as the only direct join. Now, in the visualization workspace, selecting web pages in the TreeView also selects and highlights all the hits to those pages in the scatter plot, causing them to stand out from the rest of the hits. Similarly, selecting hits in the scatter plot will select and highlight the hit pages in the TreeView, enabling users to discover which pages were hit during certain dates or times of day. Alternatively, coordinating to the scatter plot's "load" action would have a different effect. Selecting pages in the TreeView would show only those hits in the plot, filtering out all others.

We then add the capability to visualize referring sites (bottom two components). The Graphical-Table view and web-browser components at the bottom display data about external web sites that refer readers to the site shown in the TreeView. A coordination is dragged from the "select" port of the TreeView to the "select" port of the Graphical-Table. The system determines that there are two double-joins in the data schema that associate URLs to Referrers, and displays a popup menu giving the user a choice of either alternative (Figure 46). Since we are interested in only those referrers which actually resulted in hits to URLs and not all possible referrers, we choose the URL-Hits-Referrers path. We can then browse back and forth between pages in the TreeView and external sites. A final additional coordination enables us to select a referrer in the Graphical-Table View to display the referring page in the bottom web-browser. The example shows that many readers discover the website's Fusion page through Google.

### 4.3 Datafaces (Database + Interfaces)

Since Visualization schemas are based on relational database schemas (Figure 41), it seems natural to make the relationship between them obvious to the user. Datafaces (Figure 48) allows for visual connection between visualization schemas and data schemas. This facilitates easier understanding of the model and reduces the learning time considerably. Datafaces also allows simultaneous manipulation of data schema and visualization schema and hence reduces the development time for the desired visualization.

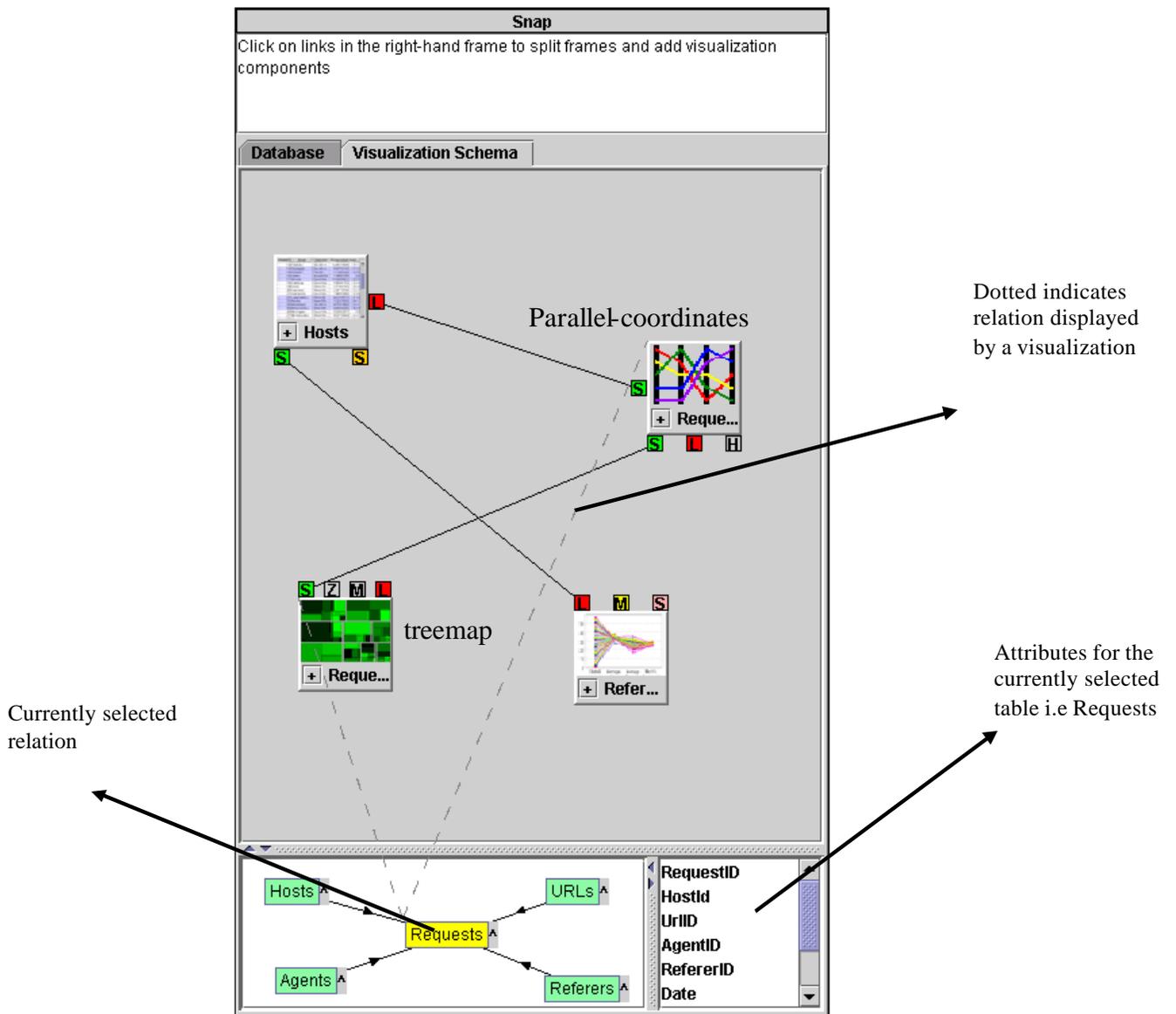
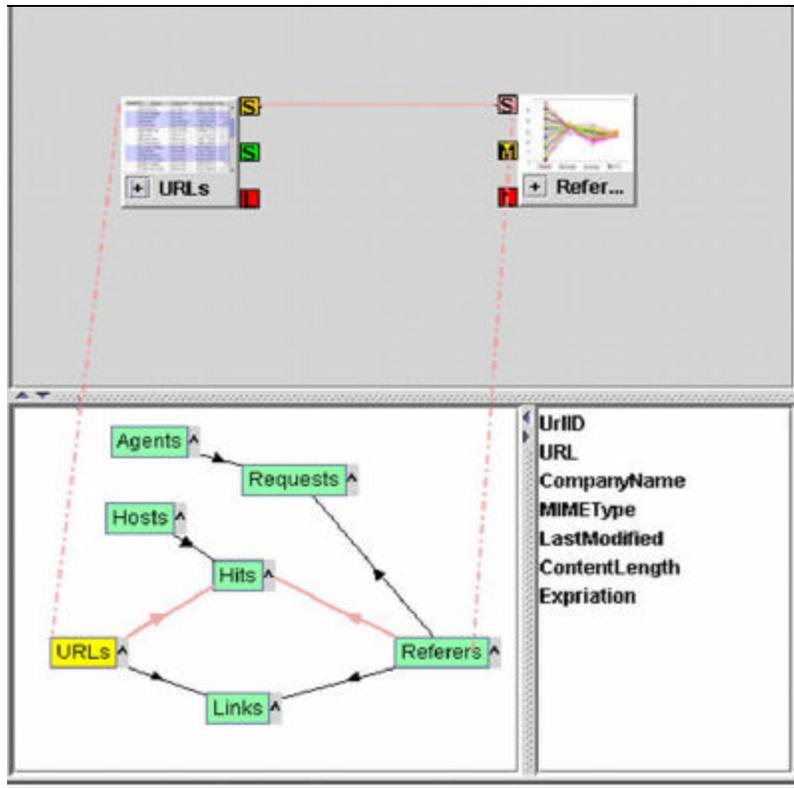


Figure 48: Datafaces: showing that both the parallel-coordinates and the treemap [39] encapsulate the Requests relation.



**Figure 49: Datafaces displaying a joinpath in the database schema corresponding to a coordination selected in the visualization schema**

Visualization schema is tiled above the database schema to emphasize that it builds on top of database schema. Relationship between relations in database schema and nodes or visualization components is shown by means of a dotted line connecting the visualization node and the relation. The actual visualization is also highlighted in the workspace. Selecting a relation in the database schema or a node in the visualization schema causes this dotted-line to appear. This allows the user to see which other views are encapsulating the same relation. User can select a coordination and see the joinpath highlighted in the database schema (Figure 49). This allows the user to visually understand the database relationship to which the coordination corresponds.

#### **4.4 Event Feedback**

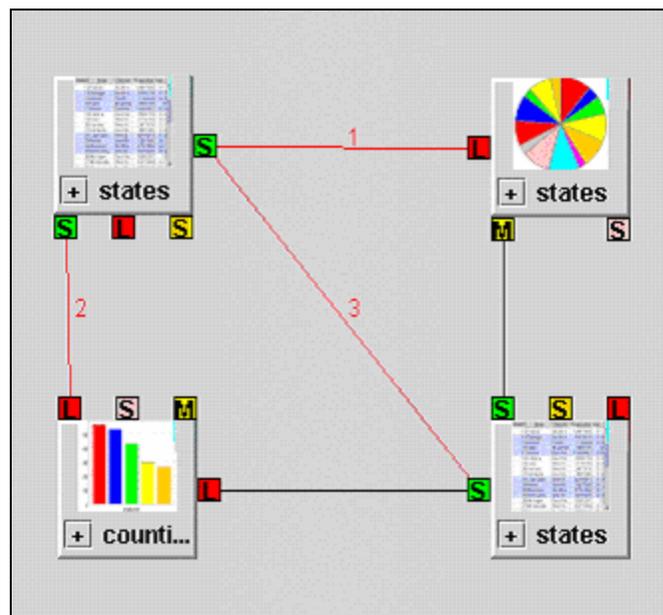
It is important for the user to know which views were affected after a direct user action on a visualization component and what the effect of the event was. Visualization schema enables this by providing feedback after every event showing which coordinations, views, and actions took part in event propagation and in which order. This feedback helps users in understanding the visualization and the coordination mechanism.

It is important to have a consistent and natural way of resolving cycles in visualization schema coordination graph (Chapter 5). The manner in which cycles are resolved should map directly to a users' mental model. For every single direct action no visualization should perform more than one direct or indirect action. We use a breadth-first mark-and-sweep algorithm to resolve cycles. The root is the node corresponding to the visualization on which the user performed a direct action. Event feedback helps users in better understanding of the cycle resolution strategy.

All traversed links are highlighted in the visualization schema. A unique order identifier is displayed next to the link to indicate the order in which the links were traversed by the coordination manager during event propagation (Figure 51).



**Figure 50: Visualization Schema Event Feedback showing the result of a “select” action on the Table visualization component encapsulating states relation (left). Two coordinations (colored) were used for event propagation. The order of coordination processing is indicated next to the coordination**



**Figure 51: Visualization Schema Event Feedback showing cycle resolution as a result of “select” action on the Table visualization component encapsulating states relation (top left). Three coordinations (colored) were used for event propagation. The order of coordination processing is indicated next to the coordination.**

## 4.5 Database Schema UI

### 4.5.1 Database Connection Interface

Connecting to a database is the first step in constructing a coordinated multi-view visualization using Fusion. User can connect to remote databases, local databases, or both. Fusion allows for connection to multiple databases simultaneously. The User Interface for connecting to databases (Figure 52) allows for creation of a list of local and remote databases to which the user wants to connect. Users can connect to these databases and then modify the database schema as per their needs.

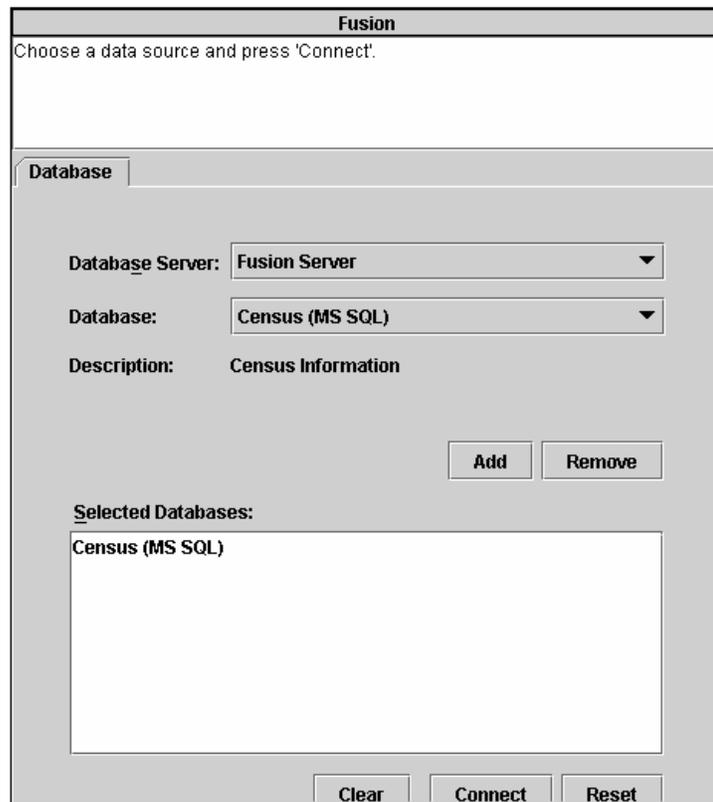


Figure 52: Database connection user interface. User can connect to multiple remote or local databases simultaneously.

### 4.5.2 Database Schema UI

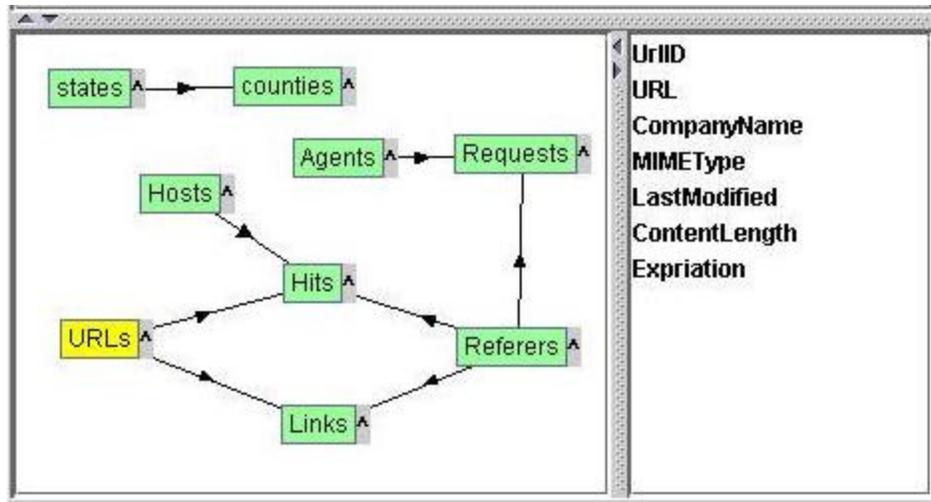
Database schemas are represented as a graph, and support direct manipulation. Nodes in the graph represent relations in the database. Edges represent the database relationships between relations. We present essentially two views of the database schema. We present an overview (Figure 53) and a detailed view (Figure 54). Using the details view users can

- Create virtual tables
- Remove existing tables
- Create virtual relationships
- Remove existing relationships
- Create virtual columns
- Delete virtual columns

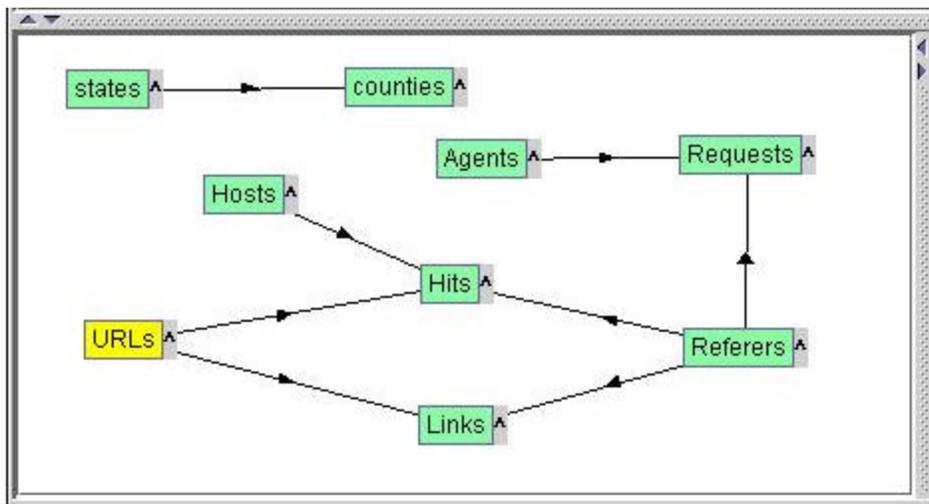
This allows users to create a custom schema which they can use to explore the data. User can return to the details view at any time from the overview view. Changes made to the schema will take place immediately and seamlessly at runtime. We would in the next two sub-sections have a look at the overview and the details view in detail.

#### **4.5.2.1 Overview view**

The overview of the data schema is shown below the visualization schema (Figure 53). The nodes represent relations and the edges represent database relationships. The cardinality is shown with the help of a triangle on the edge with the triangle pointing to the *many* side. Users can drag all the attribute of a table to the visualization schema node by dragging the '^' symbol at the right hand side of each node (see Datafaces (Database + Interfaces)). They can also choose which attributes to drag from the attribute list and then drag only those attributes. The attribute list displays all the attributes for a selected visualization. In Figure 53 the URLs relation is selected and its attributes are shown in the attribute list. User can also hide the attribute list or the graphical overview or both by using the frame separators.



(a)



(b)

**Figure 53: Database Schema Overview-view. (a)The attribute list shows the attributes for the select table, i.e., URLs.(b) Database schema with attribute list minimized/hidden.**

#### 4.5.2.2 Details view

The details view shows all the tables for all the connected databases. Each relation is represented as a node and the relationship with other relations is shown as a link (Figure 54). The link is from one attribute to the other and hence shows which attributes the relationship is based on. Users can create virtual relationships between two attributes of separate relations by simply dragging one attribute onto the other. Users also can create temporary virtual tables using SQL Direct, an interface which allows you to write SQL queries to the database (Figure 55). Any changes done to the schema take place immediately.

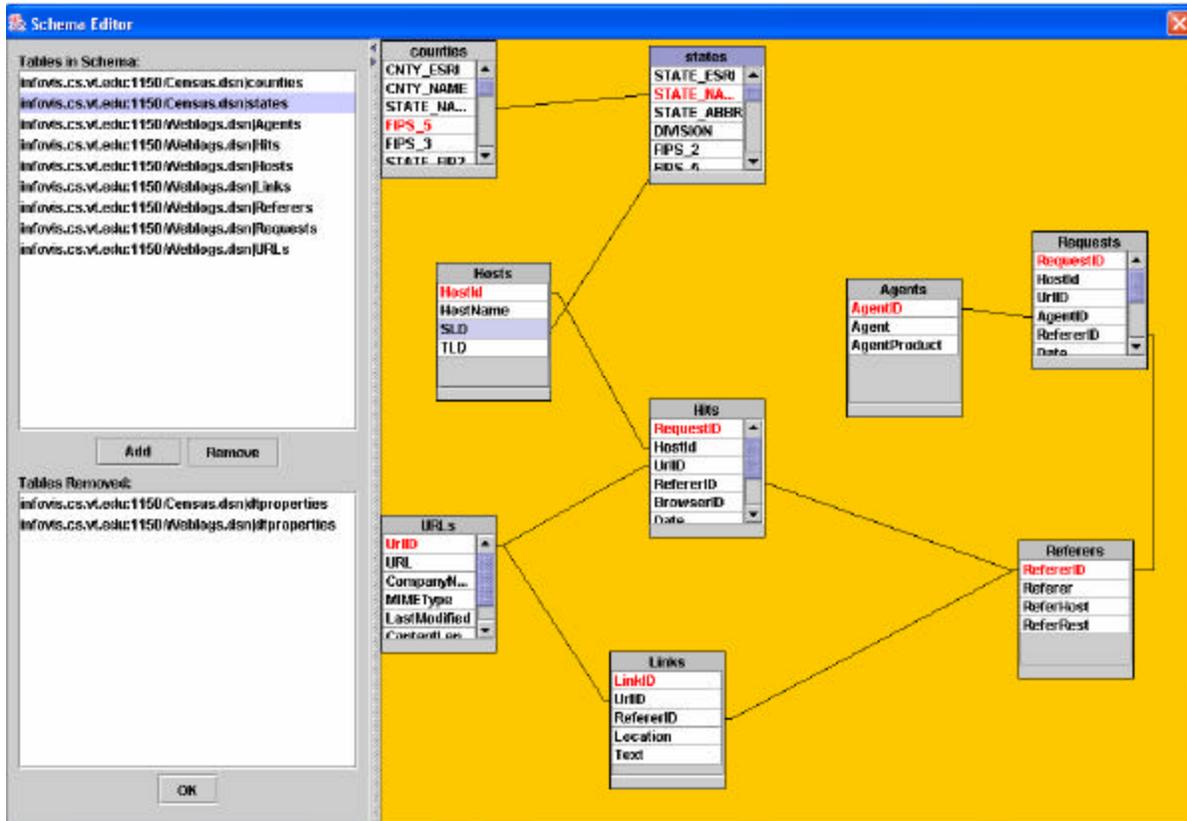


Figure 54: Database Schema-details view. Editable database schema at the right and list of deleted and available tables on the left. Database schemas for two separate databases (*Census and Weblogs*) are shown on the right.

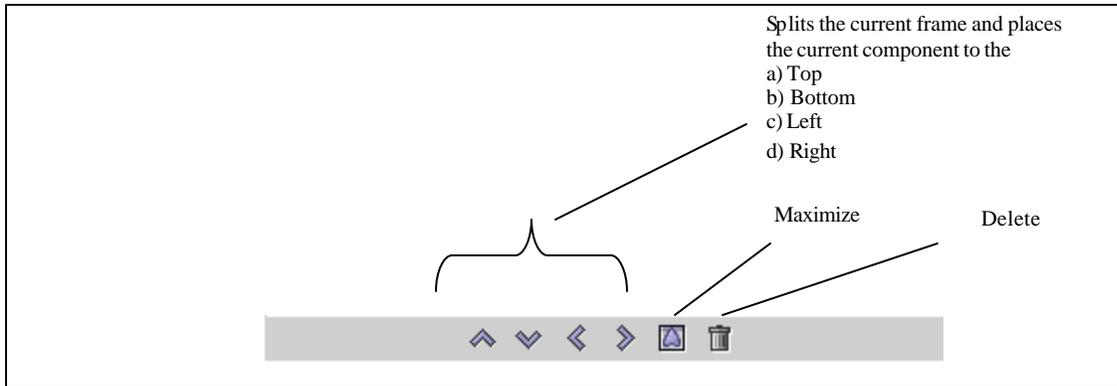
The screenshot shows an 'SQL Editor' window for 'Weblogs (MS SQL)'. The query entered is 'select \* from hits'. The result is displayed in a table with columns: RequestID, HostID, URID, RefererID, BrowserID, Date, Time, Status, and Bytes.

RequestID	HostID	URID	RefererID	BrowserID	Date	Time	Status	Bytes
1	4810	965	1719	803	9/24/1998	9:45:58 AM	OK	3534
2	2650	85	1803	361	9/24/1998	9:46:18 AM	OK	7438
3	2876	967	1337	430	9/24/1998	9:46:25 AM	OK	2134
4	5739	265	1194	276	9/24/1998	9:47:58 AM	OK	44599
5	3646	85	2643	284	9/24/1998	9:53:37 AM	OK	7438
6	3646	944	1651	284	9/24/1998	9:54:32 AM	OK	5367
7	3646	371	1688	284	9/24/1998	9:58:08 AM	OK	37799
8	3646	967	1651	284	9/24/1998	10:02:37 AM	OK	2134
9	1961	836	2484	643	9/24/1998	10:08:46 AM	OK	8332
10	4289	934	25	838	9/24/1998	10:14:47 AM	OK	3515
11	5640	85	1881	583	9/24/1998	10:15:20 AM	OK	7438
12	5315	836	2484	746	9/24/1998	10:16:53 AM	OK	8332
13	5315	85	1539	746	9/24/1998	10:17:20 AM	OK	7438
14	5073	969	1742	793	9/24/1998	10:21:26 AM	OK	98828
15	3817	266	25	643	9/24/1998	10:24:45 AM	OK	7641

Figure 55: SQL Direct. Supports running SQL queries directly against the underlying databases. Advanced users can use this for creating temporary tables and views.

## 4.6 Workspace User Interface

Fusion uses frame based approach to display different visualization components. Users can split the resize frames to create a desired workspace layout. Users can use the workspace-window-management task bar (Figure 56) to create new frames and move the current visualization to a different frame.



**Figure 56: Workspace window management task bar. This appears below every visualization component in the visualization workspace**

A visualization selection menu is displayed in each frame and the user can select the visualization component from the menu to load the desired visualization. Frame can further be split into sub frames to allow more visualizations to be loaded ( Figure 57). This allows users to create more frames as they do their data analysis.

STATE_ESRI	STATE_NA	STATE_AB	DIVISION	FIPS_2	FIPS_5	STATE_AL	AREA	POP1995	POP
1	Washington	WA	Pacific	53	53000	49	20.75	5,430,940	4.9
2	Montana	MT	Mountain	30	30000	27	45.132	870,261	7
3	Maine	ME	New Engla...	23	23000	20	9.571	1,241,362	1.2
4	North Dakota	ND	West North...	38	38000	35	21.874	641,367	6
5	South Dak.	SD	West North...	46	46000	42	22.598	729,034	6
6	Wyoming	WY	Mountain	56	56000	51	27.966	480,184	4
7	Wisconsin	WI	East North...	55	55000	50	16.477	5,122,871	4.9
8	Idaho	ID	Mountain	16	16000	13	24.381	1,163,261	1.0
9	Vermont	VT	New Engla...	50	50000	46	2.794	584,771	5
10	Minnesota	MN	West North...	27	27000	24	25.577	4,809,548	4.5
11	Oregon	OR	Pacific	41	41000	39	29.167	3,140,585	2.9
12	New Hamp.	NH	New Engla...	33	33000	30	2.677	1,140,253	1.1
13	Iowa	IA	West North...	19	19000	16	15.893	2,841,764	2.7
14	Massachu.	MA	New Engla...	25	25000	22	2.309	6,073,550	6.0
15	Nebraska	NE	West North...	31	31000	28	21.606	1,637,112	1.5
16	New York	NY	Middle Ata...	36	36000	33	13.875	18,136,061	17.9
17	Pennsylvan.	PA	Middle Ata...	42	42000	39	12.55	12,071,842	11.9
18	Connecticut	CT	New Engla...	9	9000	7	1.392	3,274,662	3.2
19	Rhode Isla.	RI	New Engla...	44	44000	40	0.293	989,794	1.0
20	New Jersey	NJ	Middle Ata...	34	34000	31	2.057	7,946,299	7.7
21	Indiana	IN	East North...	18	18000	15	9.932	5,803,471	5.5
22	Nevada	NV	Mountain	32	32000	29	29.969	1,530,109	1.2
23	Utah	UT	Mountain	49	49000	45	23.067	1,954,899	1.9

(a) Before Splitting the frame

(b) After splitting the frame. Left frame shows the Visualization Menu

Figure 57: Frame splitting. (A) Shows a table loaded in a frame. (B) Shows the same table on the right hand side after splitting. A blank frame is created on the left side which displays the visualization menu.

## 4.7 Discussion

### 4.7.1 Advantages

Visualization schemas have many advantages. Its direct manipulation approach enables users to construct and modify multiple-view visualizations very rapidly. This reduces development time for a custom multi-view visualization. Because it closely resembles relational data schemas, learning time is greatly reduced from the previous form-based approach [32]. Also, visualization

schemas are used to specify user interactions, which are a level that users are familiar with, in contrast to dataflow specifications which are at a much more detailed data processing level.

Furthermore, it provides a visual overview of the coordination structure of the visualization, which helps users quickly learn how to operate the visualizations and understand its response. This helps to solve a long-standing usability problem with multiple-view visualizations in general. Hence, visualization schemas improve usability at both the construction phase as well as the operation phase. Visualization schemas also can provide a compact representation of visualizations for the purpose of browsing and recognizing many alternate designs. It also provides a context in which to show other information related to visualizations and their use, such as animations of propagation of actions through the coordination graph or saved sequences of actions such as macros. Finally, it enables a new level of flexibility in visualization that matches that of databases. It does not require programming, which enables visualizations to be rapidly updated along with dynamic database schemas.

Datafaces demonstrates the relationship between Visualization Schema and database schema. It aids in better understanding of the model and hence reduces the learning time considerably. It allows the user to quickly see visualizations that are encapsulating the same relation. This knowledge can be used to coordinate these visualizations in brushing type coordination. Datafaces facilitate simultaneous manipulation of data schema and visualization schema. This further reduces the development time for personalized multi-view visualizations.

## **4.8 Future Work and Improvements**

Visualization Schemas provide the foundation for several extensions.

### **4.8.1 Saving Schemas**

Support for saving schemas, both visualization and database, so that they can be shared, packaged and sent to others or accessed remotely. This will allow users to share their findings with other and to enable others to examine and benefit from results. The format for saving the schema should be in human readable form so that humans can understand and manipulate the schema structure. Text based approaches or XML would work well for the purpose (Figure 58).

Users also shall be allowed to construct, view, edit, save, share, reuse, and replay interaction sequences. Another approach could be to save the schemas in the form of specification language defined in Chapter 3. A sequence of coordinated pair specifications can be saved along with database connection information. Saving the layout of visualization components in the visualization workspace and saving the state of individual visualization components would require extra parameters in the format. Users will be able to publish their custom visualizations on the web for others to analyze and observe.

```

<?xml version="1.0" encoding="UTF-8" ?>
- <java version="1.4.1_01">
- <database>
  <server>snap.cs.vt.edu</server>
  <name>Census(MS SQL)</name>
  <user>snap</user>
  <passwd>snap</passwd>
</database>
+ <void property="snapLayoutManager">
- <void property="Relationships">
- <object class="java.util.Vector">
  - <void method="add">
    - <object class="snap.Relationship">
      <id0>snapwindow0</id0>
      <action>select</action>
      <id1>snapwindow1</id1>
      <action>select</action>
      <joinpath />
    </object>
  </void>
</void>
- <void method="add">
  - <object class="snap.Relationship">
    <id0>snapwindow1</id0>
    <action>select</action>
    <id1>snapwindow2</id1>
    <action>select</action>
    <joinpath />
  </object>
</void>
</object>
</void>
</java>

```

**Figure 58: Sample XML File for saving and restoring Schema information. Information is stored in three distinct sections (a) database connection information, (b) Workspace and layout information, and (c) Relationship between visualization components**

#### 4.8.2 Bookmarks

Support for book-marking a visualization state as it is being developed so that a user can return back to the saved state could be added to Visualization schemas. Snapshots of system state can be taken in the same format in which the visualizations are saved (Figure 58).

### **4.8.3 Collaboration**

Support for collaboration in construction of multi-view visualization can be added. Users would share visualization schema, and database schema. There shall be a unique shared connection to the database. Shared visualization schema could act as the means to achieve collaborative visualization development.

### **4.8.4 Feed Forward**

Some database management systems allow the user to see what tables and what indexes would be used for a query before the query is actually executed. This concept can be used in visualization schema to show which visualizations will be updated if the user performs a direct action on a visualization component. This feed forward would allow users to choose the most effective actions for their tasks.

## **4.9 Summary**

This chapter presented the user interfaces for Data Schemas and Visualization Schemas along with the concept of Datafaces. Visualization Schema user interface enables users to explore information by quickly constructing coordinated-visualization interfaces without programming. Users first open relations into visualizations, then coordinate them by selecting actions to tightly couple. Data schemas and visualization schemas are both represented as graphs supporting direct manipulation. Datafaces combines data schema and visualization schema to enable users to simultaneously manipulate data schema and view-coordinations. Together, the concepts of Datafaces and visualization schemas aims at supporting flexible, dynamic, user-guided construction of multi-view visualizations for diverse databases.

## Chapter 5 Visualization Schema: Architecture

### 5.1 Overview

Snap software architecture is event-based, implicit invocation architecture [38]. Component developers are concerned only with firing and receiving events and not the coordination architecture. Coordination Manager utilizes the database's relational schemata to translate events between visualizations that wrap related data [33]. When a view is added, The Snap architecture consists of three major layers for coordinating components (Figure 59).

Lack of support for coordination across compound joins meant that users had to break a compound join into a sequence of simple joins and then create an intermediate view for each simple join. Intermediate views resulted in unnecessary complication of coordination structure. For example, in coordinating views encapsulating *URLs* and *Referrers* (Figure 42), users would have to create an intermediate view encapsulating either *links* or *hits relations*.

In this chapter, we discuss critical architecture changes to support multiple databases and event propagation across compound joins. As shown in Figure 59, coordination manager, coordination graph, and the database schema were changed to add support for multiple databases and event propagation across compound joins.

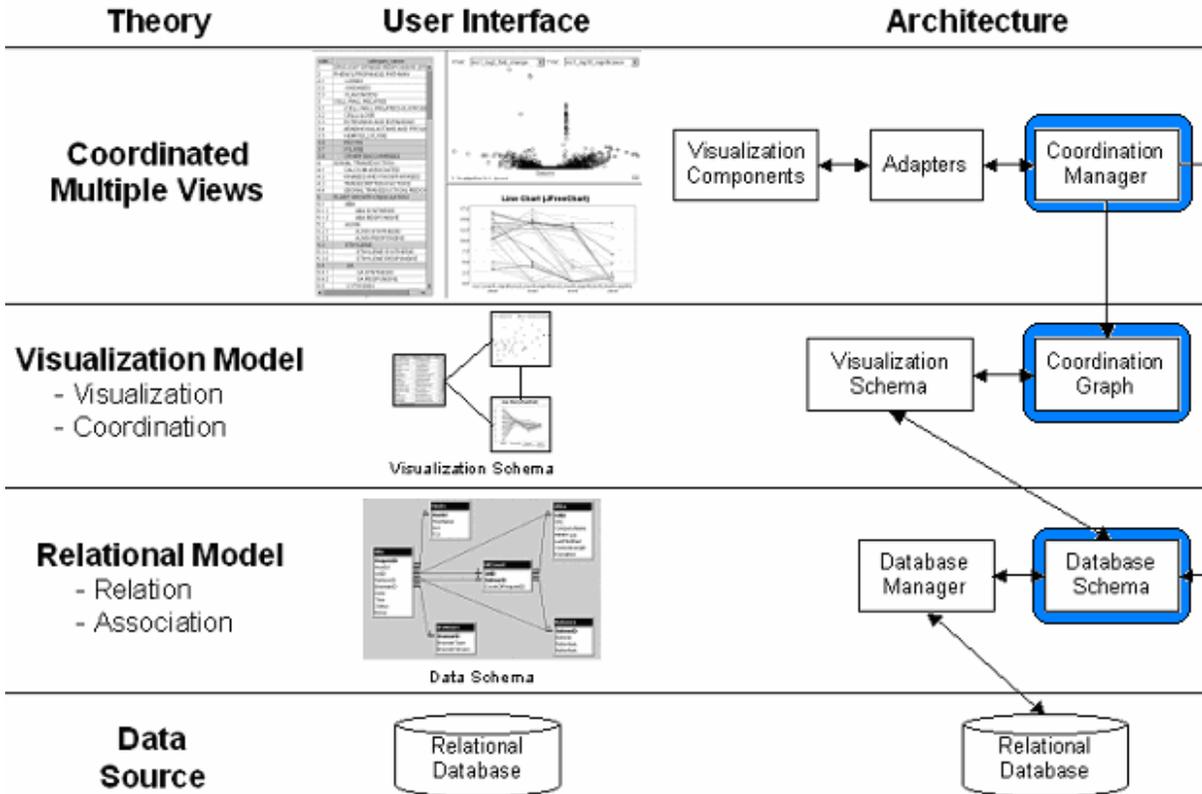


Figure 59 – Web-based Snap Architecture [9] indicating the separate layers needed to implement the Snap system. The data source maintains the data to be visualized. The first layer deals with providing connectivity to data sources and describes join associations between the separate data relations. The second layer supports the assignment of data relations to visualization components, and the coordination of events between components. The third layer handles all communication with the visualization components, and is responsible for the receiving and firing of events. Coordination manager, coordination graph, and database schema were changed to add support for multiple databases and event propagation across compound joins.

### 5.1.1 Previous Snap architecture

The previous Snap architecture supported a static group of visualization components and required a large install base with each component installed on the machine [9] [27]. It supported only single tuple actions and did not utilize data associations when propagating events. The group of visualization components that could be used was static and support for connecting to multiple databases was lacking. This initial architecture was extended to a run-time extensible architecture allowing visualizations components to be dynamically added to the system [30]. Support for multiple databases and coordination across compound joins was still lacking. Users have to rely on use of unnecessary intermediate views to coordinate views across compound joins.

We will discuss further extensions and enhancements to the Snap Architecture. Key features and enhancements include support for joinpaths, multiple databases, better key translation, virtual relationships, virtual tables, and new adapters. [29] and [9] provide detailed discussion on the Snap architecture.

### 5.2 Coordination manager

The Coordination manager is the core component of Snap which manages the interaction between various components. The Coordination manager uses the coordination graph, key converter, and database schema to propagate events across database joins (Figure 59). Information about coordination structure is stored in the coordination graph (Figure 60).

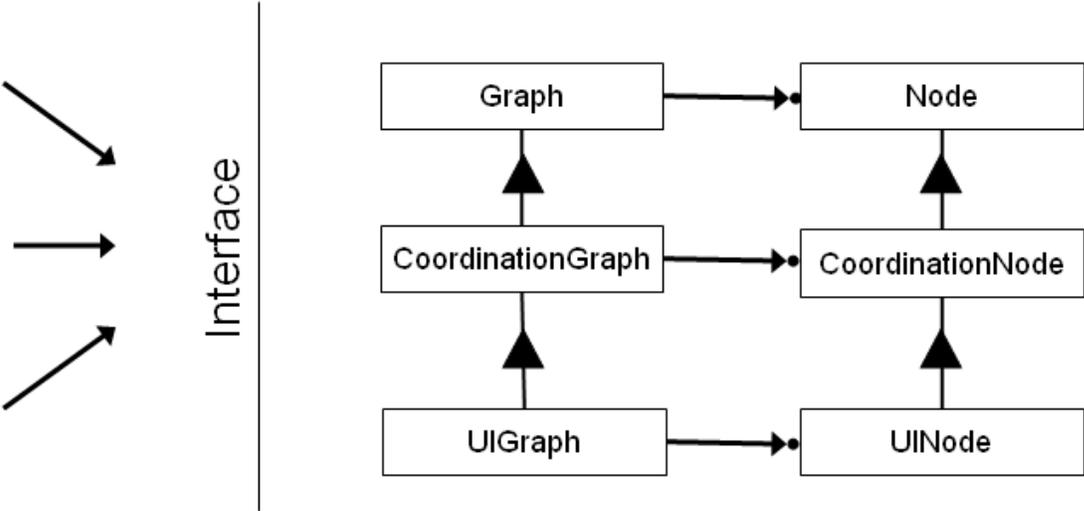
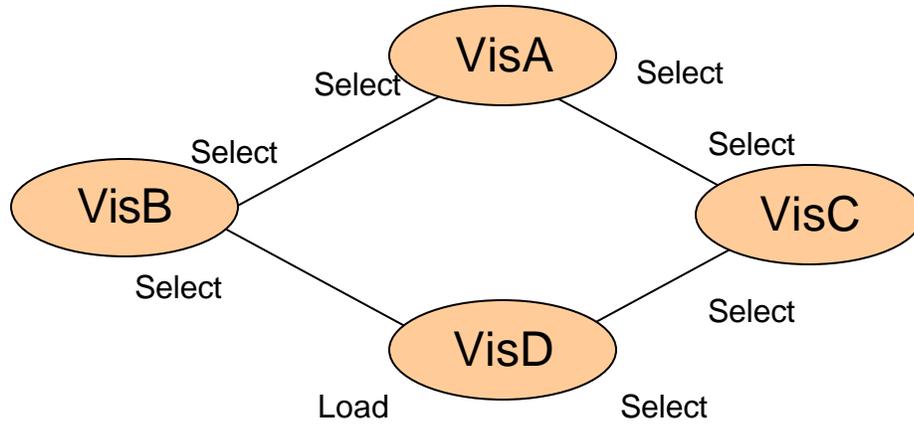


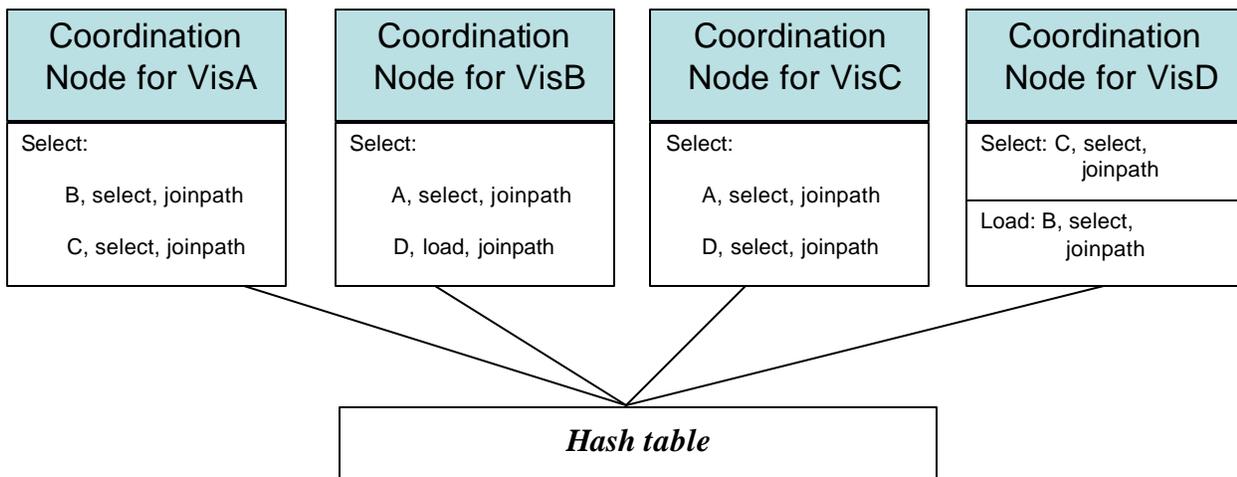
Figure 60 – Coordination Graph class diagram.

#### 5.2.1 Coordination Graph

The Coordination Graph is a collection of Coordination Nodes; each Node aggregates a visualization. The Coordination Graph maintains an interface that supports the Coordination Manager’s mark-and-sweep implementation of event firing. The UIGraph and UINode provide additional functionality needed by the user interface. Since both the coordination manager and UI can access the graph structure, a semaphore-like synchronization is implemented.



**Figure 61: Example coordination graph with four coordinated visualization components.**



**Figure 62: Coordination Graph Data Structure showing coordination nodes for each of the visualizations represented in Figure 61. Each node stores a hash table with actions as hash keys.**

**The Coordination graph has been modified to store the joinpath along with the visualization information in the nodes (**

**Figure 62).** Each node maintains a hash table for storing its connections. The hash table is keyed by the actions supported at the node. The value is a list of Node-Action pairs that are connected to this action. The structure of this hash table can be shown as:

Hash table.Key -> List of Node-Action pairs

This can be demonstrated using (Figure 61) and (**Figure 62).** The node for VisA would contain a hash table with an item associated to its “Select” action. Its value is shown in the following list:

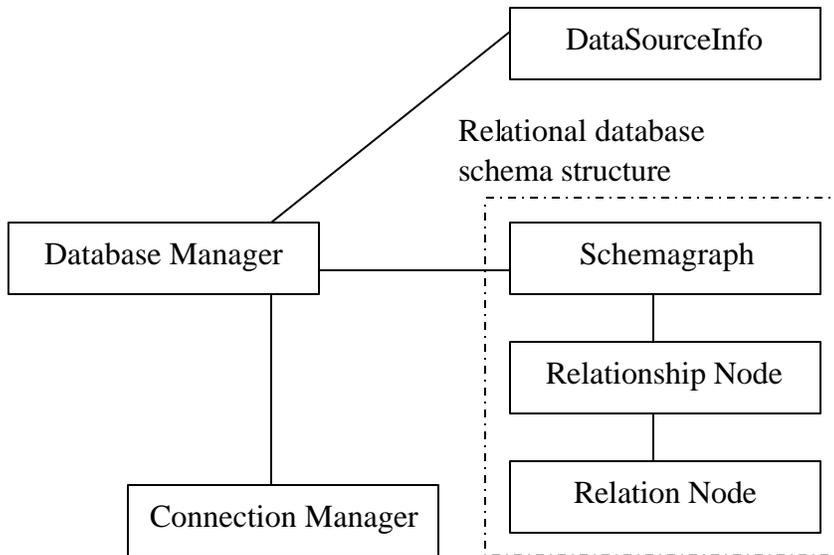
```
VisA."Select"-> {  
    (VisB,"Select", joinpath),  
    (VisC,"Select", joinpath)  
}
```

Snap is responsible for coordinating actions between the visualization components. As the user configures the visualization schema at the user interface the coordination graph is constructed at the back-end. The database relationship, i.e., joinpath is stored as a part of the coordination and this facilitates changing of joinpath by the user dynamically at runtime. The Coordination Manager uses a breadth first mark and sweep algorithm to resolve cycles in the coordination graph. This algorithm ensures that each coordinated visualization receives the appropriate event only once [29].

### **5.3 Integrated multiple data sources**

Fusion supports concurrent connections to multiple databases. For each connected database, we store the information about the database schema, i.e., number of tables, relationships, primary keys, and foreign keys. Each database is uniquely identified by its fully qualified name which involves server name, port number (if applicable), and DSN name. A pool of connections is maintained by the database manager and each connection is registered in this pool. Access to connections is synchronized to avoid conflicts and race conditions.

Behavioral differences across database types are classified as static members for DataSourceInfo, a class that encapsulates data source information. DataSourceInfo provides default values for connecting to different databases like SQL, Oracle, MySQL, Postgres SQL, etc. Query class is used to encapsulate JOIN and WHERE filters. A query object does not span multiple databases. Multiple database support is accomplished by using multiple Query objects. The ConnectionManager keeps track of multiple database connections.



**Figure 63: Database Manager uses dataSourceInfo, schemagraph connection manager. SchemaGraph uses Relationship node to store relationships between relations. Connecton manager manages various database connections while DataSourceInfo stores database specific information for different type of databases.**

## **5.4 Database schema**

Fusion requires understanding of database relationships in order for the user to coordinate and propagate events between two views. Database relationships are retrieved from the database and stored in a local data structure. This structure provides an interface to visualization schema and coordination manager and is maintained by the database schema subsystem [9].

### **5.4.1 Relational database schema structure**

The relational database schema structure stores information about relation's, attributes, primary keys, foreign keys, and relationships. The Relationship node and Schemagraph are used to represent the structure (Figure 63). The Relationship node stores information about the relationship between two relations, while schemagraph uses Relationship node and is an adjacency list graph representation of all relationships within the database. Database Schema exposes the standard API (Figure 64). Database Manager provides an interface for querying the database (Figure 65).

```

getFields(String tableName)

getJoinPaths(String tableFrom, String tableTo)

getPrimaryKeys(String table)

getRelation(String table1, String table2)

getTables()

```

**Figure 64: Database Schema interface to the structure.**

```

translateEvent(JoinPath joinpath, Vector inputs)

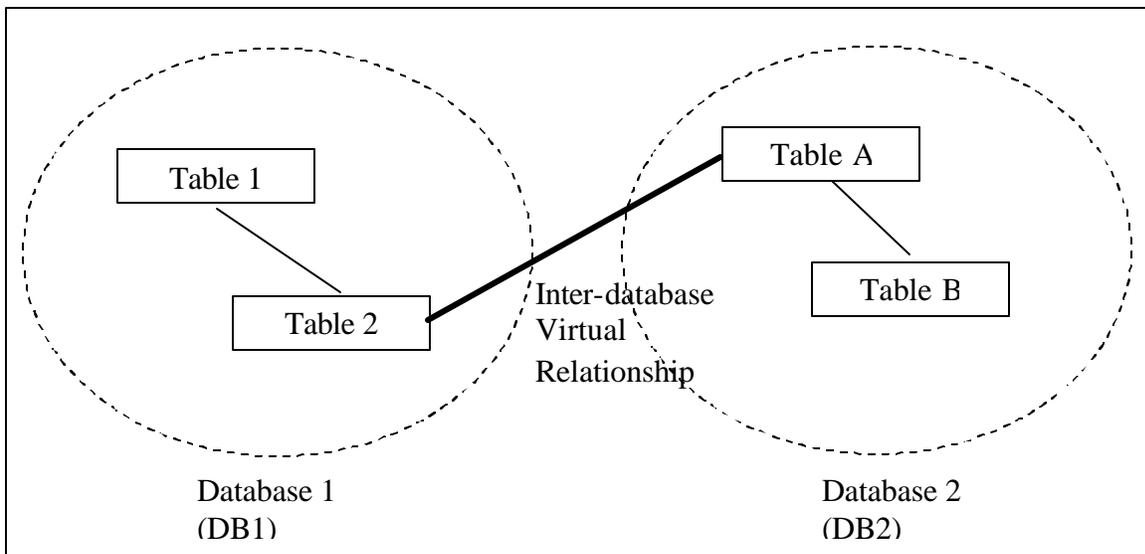
getResultSet(Query sql)

```

**Figure 65: Database Manager interface to access Data.** [translateEvent](#) method converts the input Keys from a source table into output Keys in a destination table using specified joinpath. [getResultSet](#) method Gets a resultset of data for the specified query

## 5.4.2 Virtual Database Schema

To support interaction between multiple databases it is essential that we allow relationships to be created across databases. Since these relationships do not exist in the database these are called virtual relationships (Figure 66).



**Figure 66: Virtual relationship.** A virtual relationship can exist within the same database (Intra-database) or across databases (Inter-database).

To support the creation of virtual relationships, virtual tables, and virtual columns Fusion builds a virtual DatabaseSchema structure (Figure 67) which is built on top of database schemas for individual databases. Fusion combines individual database schemas into one virtual database schema and it is this virtual schema that is used by other Fusion components. Building this virtual schema allows users to configure and build custom database schema. Users also can edit existing database relationships. Fusion provides a user interface for users to do modifications to the Virtual database schema (Figure 68).

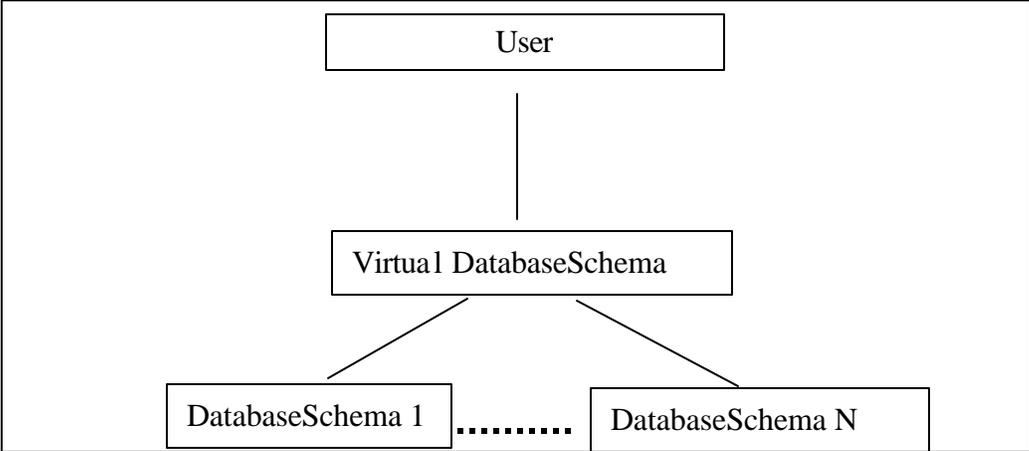


Figure 67: Virtual Data Schema builds on top of data schema’s for individual databases. User can create virtual tables and relationships in the virtual schema.

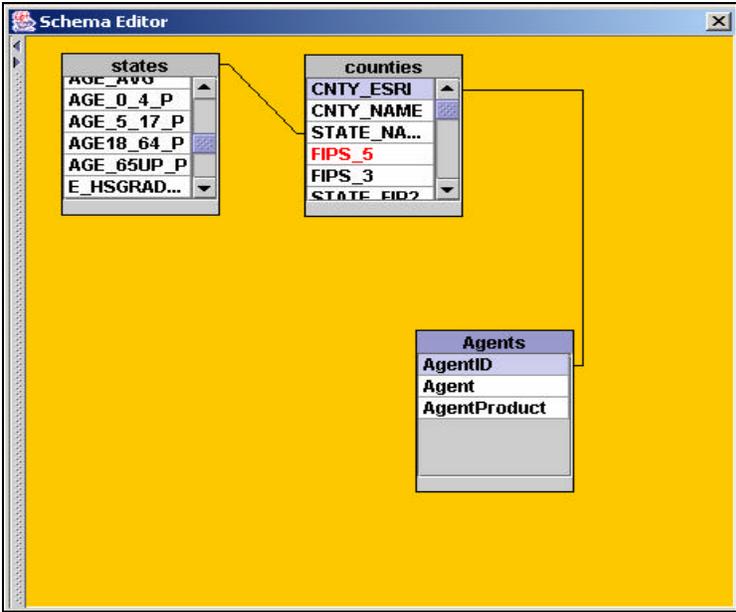
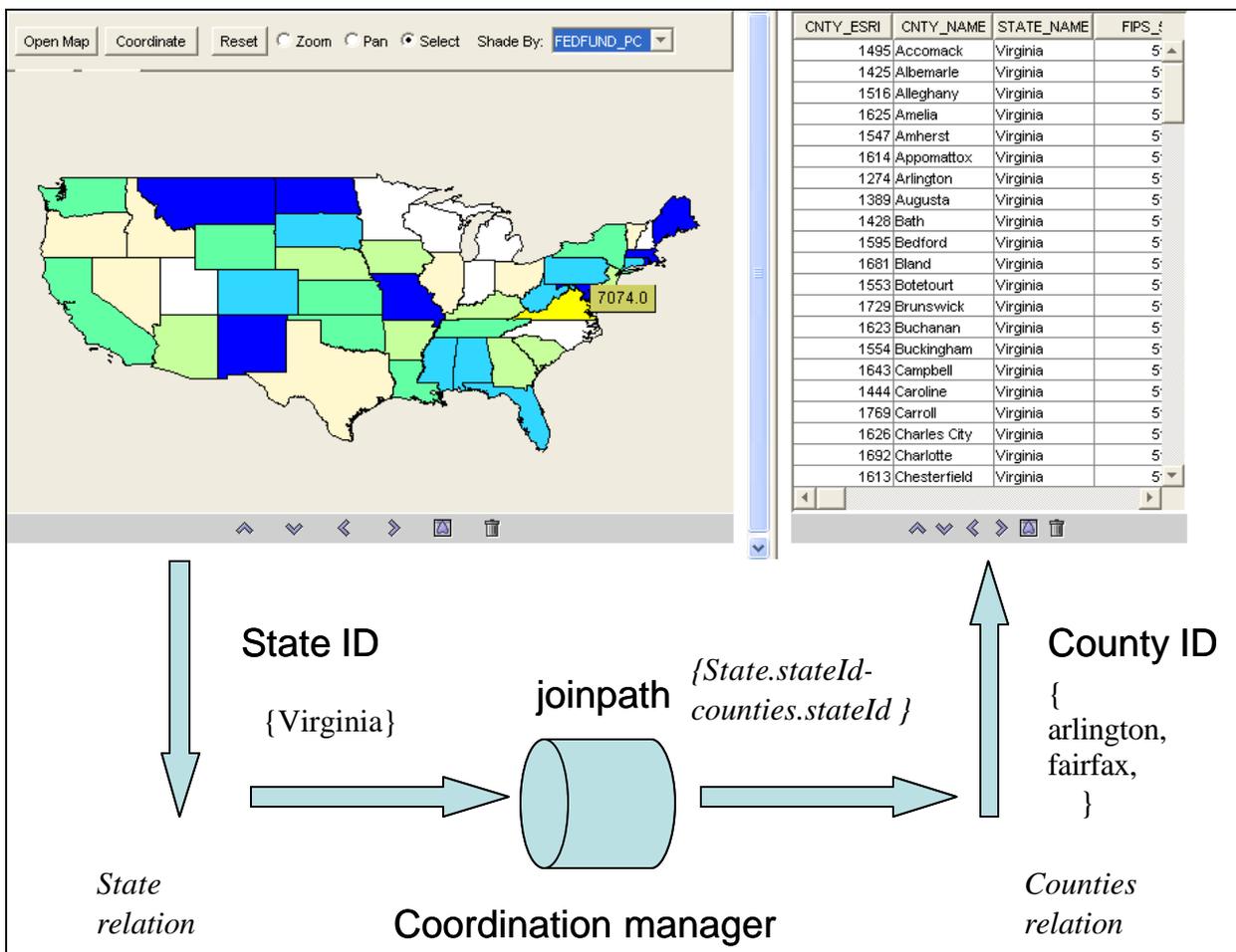


Figure 68: Virtual data schema editing tool. Users can create custom data schema to be used by Fusion for component coordination and event propagation.

## 5.5 Event Translation & Propagation

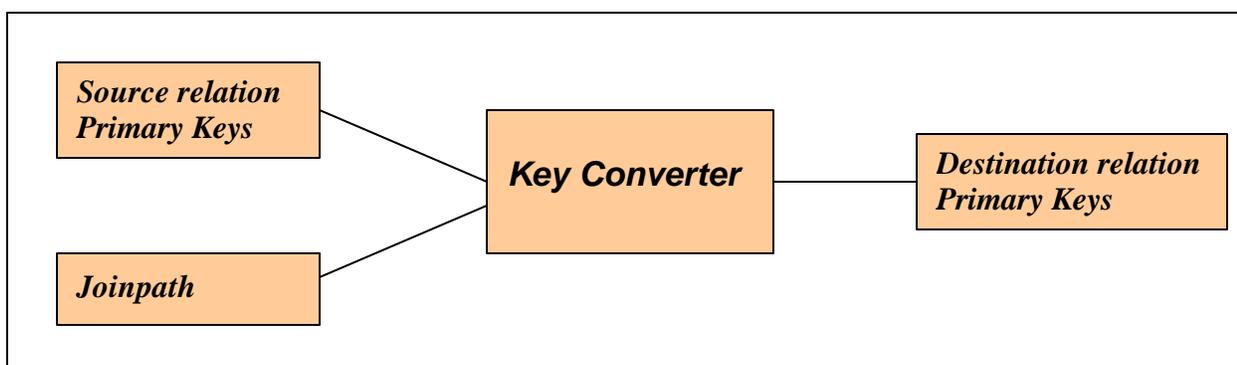
When coordinating events between two components that encapsulate different relations, event translation and key conversion is needed to join the relations [30]. For example, in Figure 69, Map fires events on *state id* whereas table expects to receive *county ids*. Key conversion is needed to convert *state id* keys to appropriate *county id* keys. The Coordination Manager utilizes the data schema to translate events appropriately based on the underlying data join associations. The Coordination manager uses the Key Converter (Figure 70) component for translating the source keys to appropriate destination keys. The Coordination Manager acts as a Mediator [13] between the individual visualization components.



**Figure 69: Key conversion.** Coordination manager uses key converter to convert keys from source relation to destination relation. Map and Table are related by a *select-to-load* relationship. Map fires events on *state id* whereas table expects to receive *county id*. Coordination manager converts *state id* key values to corresponding *county id* key values based on *joinpath*.

### 5.5.1 Key Converter

Key converter converts the primary keys for one relation to primary keys of another relation based on database join relationships. Whether key conversion is needed or not depends on the kind of database relationship that the two relations share. If the two relations share the primary keys (e.g., same relation in separate views) then no key conversion is needed but if two relations do not share primary keys then key conversion needs to be performed as an intermediate step to find which tuples in the destination relation are related to the ones in the source tuples (Figure 69). The database relationship is extracted from the virtual database schema.



**Figure 70: Key Converter.** Converts source relation primary keys to destination table primary keys based on joinpath

Key conversion is performed as a sequence of JOIN queries to be executed on the database. Each new <relation,key> element in the joinpath results in addition of another JOIN clause to be added to the statement. We make use of INNER JOINS for queries. For example, in Figure 44 selecting multiple URLs in the tree-view requires the key converter to convert keys from the URL's primary key to primary keys of the Hits and Referrer relations. If three URL tuples whose URL IDs are 4, 5, and 6 are selected, then the data schema manager constructs the following query to translate the URLs into Hits for the scatter plot:

```
SELECT Hits.RequestID
FROM Hits INNER JOIN URLs
ON Hits.UrlID = URLs.UrlID
WHERE URLs.UrlID IN (4,5,6);
```

For coordinating across intermediate relations, event translation requires a compound join query. For the table-view of Referrers, the following query retrieves the Referrer tuples associated with the selected URL tuples:

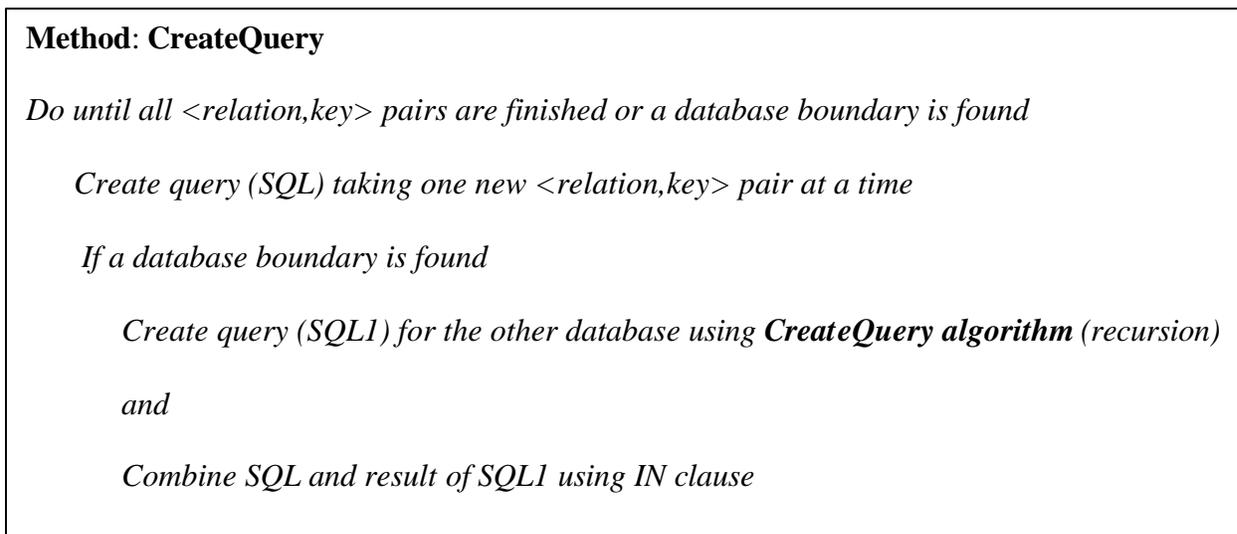
```
SELECT Referers.RefererID
FROM Referers INNER JOIN
(Hits INNER JOIN URLs
ON Hits.UrlID = URLs.UrlID)
ON Referers.RefererID = Hits.RefererID
WHERE URLs.UrlID IN (4,5,6);
```

For firing “load” actions, the query would be modified to retrieve all the attributes to be displayed in the destination component, instead of only the primary key attribute.

Query generation and event propagation for multiple databases is complicated and requires intermediate queries. Since we cannot execute a cross-database query, we execute individual single-database queries and then use their results within other queries for cross-database key conversion. Query construction is a recursive process with two distinct steps. In the first step, we create constraints, i.e., *queries* for each individual database available in the joinpath from source to destination. In the second step, we combine these queries using an IN clause. For example, if views encapsulating table1 and tableB (Figure 66) were coordinated and tuples with IDs 4, 5, and 6 are selected in table1 then the following query retrieves the tableB tuples associated with the selected table1 tuples:

```
Select DB2.tableB.primary_key from DB2.tableB INNER JOIN
DB2.tableA ON
DB2.tableB.foreignkey_wrt_tableA = DB2.tableA.primaryKey AND
DB2.tableA.foreignkey_wrt_table2 IN (Result of select
DB1.table2.primarykey from DB1.table2 INNER JOIN DB1.table1 ON
DB1.table2.foreignkey_wrt_table1=DB1.table1.primarykey And
DB1.table1.primaryKey IN (4,5,6) )
```

where the second nested query is executed separately and its result are used within the IN clause in the first query. Using this mechanism SQL queries are executed against a single database at a time. The schema of the recursive function is shown in (Figure 71).



**Figure 71: CreateQuery algorithm.**

## **5.6 Discussion**

We discuss application of Fusion in building data driven websites. We also cover some performance issues and challenges involved in integrating JavaScript based web-components with Fusion.

### **5.6.1 Visualization Schema for constructing Data Driven Web sites**

Visualization Schemas can be used for construction of data-driven websites. For example, if data about products (say Digital Cameras) is maintained in a relational database schema then developers can use Fusion to build a prototype data-driven website for sale of the product online. Programmers will coordinate web-page components with other JavaScript components using the visualization schema. They will be able to save the schema and publish it for different users to visit. The distinct advantage of using Fusion is that the relationships and components can be configured dynamically. This would allow for construction of websites which are easy to maintain and upgrade. Fusion could be used for rapidly building website prototypes and for testing. To build web-page-components which implement and understand the Fusion protocol we would need JavaScript Adapters.

## 5.6.2 JavaScript Adapter

A JavaScript adapter is needed for construction of web-page components and data-driven websites. JavaScript adapter will allow JavaScript components to work with Fusion. To work effectively with Snap the JavaScript adapter should support communication between Java and JavaScript. Netscape's LiveConnect [21] and Java's support for DOM [37] could be used for this communication. Fusion's database manager can, if needed, provide the JavaScript component with access to the database. One of the basic components that would be used for a data-driven website is an HTML loader which when provided with a URL could load the page. Some way of specifying the layout (possibly a language) of the page would also be required. Other components could include a table builder to show data, an image loader to show images, etc.

## 5.6.3 Performance issues

### 5.6.3.1 *Key conversion*

Key conversion is one of the performance bottlenecks in Fusion. Currently key conversion is done at run-time for each action ,i.e., for each user event a query is fired against the database to get appropriate keys. This could be improved by caching the primary keys and foreign keys for loaded relations. We would need the database manager to get appropriate keys based on primary key foreign key relationships already stored in the schema.

### 5.6.3.2 *Schema retrieval*

The Database Schema retrieval algorithm is  $O(n^2)$  (n being number of tables), where each pair of tables in the database schema are compared for possible relationships. A better algorithm for database schema retrieval would significantly improve the performance. Another way to address this issue would be to allow users to save database schemas to be loaded into Fusion. This will avoid extracting the schema every time Fusion starts.

## 5.7 Summary

This chapter described the critical enhancements and architectural changes to support multiple databases and event propagation across compound joins. Multiple database support is realized by means of a virtual database schema which builds on top of individual database schemas. Compound join support is realized by enhancing the coordination manager to use joinpaths as an

attribute of the coordination and use this for key conversion. Key conversion mechanisms and issues involving multiple databases were also discussed.

## Chapter 6 Conclusions

### 6.1 Contribution

This research makes the following contributions:

- *Visualization Schema Model*
  - *Enhanced Model*: An extension to the existing Snap model to generalize the concept of coordinations between components as any joinpath in the data schema. Inclusion of joinpath in coordination description enables specification of coordinations spanning multiple databases.
  - *Enhanced Specification language*: A specification language for representing multi-view visualizations is presented.
- *Visualization Schema User Interface*:
  - *Visualization Schema diagram*: Visualization schema diagrams which are visually represented similar to data schema diagrams map naturally to the users' mental model. The similarity between visualization schema diagram and data schema diagrams reduces learning time for the user.
  - *Datafaces*: The concept of Datafaces which helps users to view and modify data schema and visualization schema simultaneously. Datafaces also displays relationships between data schema and visualization schema. This helps users to comprehend dependencies and to construct desired visualizations with ease [30].
- *Visualization Schema Architecture*: Extension to existing Snap architecture to facilitate integration of new features included in the model. Some of the changes include:
  - *Diverse data integration*: Enhanced database schema to allow Fusion to connect to diverse data sources and to understand complex relationships.
  - *Key converter and Coordination manager*: Enhancements to Coordination manager and Key converter to facilitate coordination of views and key conversion based on complex database join associations and multiple data sources.
- *Implementation*: An implemented system (Fusion) that realizes the Visualization Schema model, user interface, and architecture.

## 6.2 Benefits

Visualization schema provides many benefits to developers, researchers, and user interface designers. Because of its fundamental similarity to relational data schemas, visualization schema inherit many benefits of relational database schemas.

- *Increases user base*: Visualization schema allows users familiar with relational database concept to quickly construct multi-view visualization for data analysis. Users don't have to be expert database programmers and don't need to write queries to load data into components.
- *Overview and Feedback*: Visualization Schema presents an overview of coordination structure at all time. This helps in increasing retention time. Event Feedback provided after every event demonstrates the working of the system and helps in understanding the functioning of the system and the visualization generated.
- *Easy learning*: Datafaces helps the user in understanding the visualization schemas model by showing the relationship between data schema and visualization schema to the user at all times.
- *Better Multi-view visualizations*: Users can build better multi-view visualizations as they can coordinate relations that are not immediately related without creating intermediate views.
- *Reduces errors*: Visualization schema provides a direct manipulation based interface for specification of coordinations and allocation of data to views. This maps to a users' mental model and so the user feels in control of the application. It reduces the learning time and errors thereby reducing the time needed to create a multi-view visualization.
- *Multiple diverse data sources*: Integration of multiple local and remote databases allows users to discover unforeseen relationships between different data sets. Ability to modify virtual database schema provides much needed flexibility in data analysis.
- *Prototyping*: Interface designers can use Visualization schema and Fusion for rapid prototyping. They can quickly construct a prototype and present it to the client for prototype-based requirements gathering.
- *Domain Independent*: Fusion can work with any kind of relational database. Visualization Schema and Fusion hence can be used in any domain with domain specific components.

### 6.3 *Limitations and future work*

Future work and limitations have been discussed after every chapter. We have discussed the possibility of incorporating data mining (Chapter 3), developing normalization guidelines for development of multi-view visualizations (Chapter 3). We talked about saving visualization schemas, managing bookmarks, providing feed-forward, and making Fusion collaborative (Chapter 4). We also discussed about using visualization schemas for developing data driven web sites, incorporating different technologies, and performance issues (Chapter 5).

Other limitations and future work needed to overcome them includes:

- *Database caching and pooling*: A better mechanism for caching keys and pooling database connections is needed to increase the performance efficiency of Fusion.
- *Snap testing tool*: Development of a testing tool for Fusion components would greatly help the development process for individual components. Since Fusion is a web based system the only way for component developers to test their components is to submit the component and then test it with the Fusion system. A testing tool which would present a light weight implementation of the Fusion coordination manager can be used to test components.
- *Data Transfer format*: Currently data is passed between components as *resultsets* and each component has to provide an interface for loading this resultset. It would be helpful if we use XML to pass data between components. This will allow passing data in human readable form and will help in integration of Fusion with other online tools.
- *Performance*: Because of the interpreted nature of Java, the performance is far from optimal. Java's advantage of being platform independent was the primary reason in its use as a choice of implementation language.
- *Model*: Visualization Schema model is based on the relational database model. Other forms of data (e.g., hierarchical data) needs to be converted to relational form before it can be effectively used with the system. Extending the model to support other forms of data sources and data models (such as XML) would be one of the challenges in the future. Visualization schemas can also be used to build visual interfaces (e.g. Envision [20]) for digital libraries. Communication between digital libraries and Fusion can be done using interoperability protocols like VIDI [43].

## **6.4 Conclusion**

Visualization schemas may help users, researchers, designers, and developers in rapidly constructing personalized coordinated multi-view visualizations for diverse data. This research signifies progress made towards applied information visualization and is a step towards creating an environment supporting “*Visualizations for the user and by the user*”.

# Appendix A: Scenarios

## US Census



Figure 72: Database schema showing one-to-many (1: M) relationship between states and counties relations.

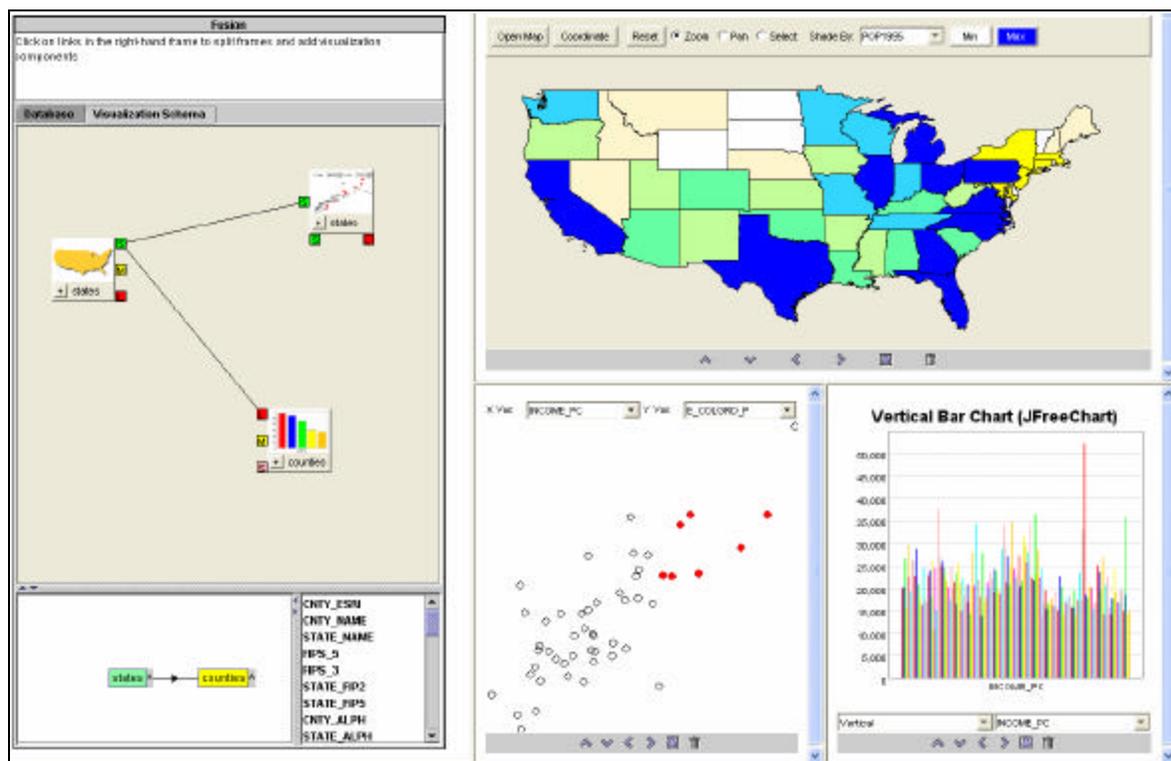


Figure 73: Multi-view visualization build using Fusion and visualization schema for Census database (Figure 72).

# Web logs

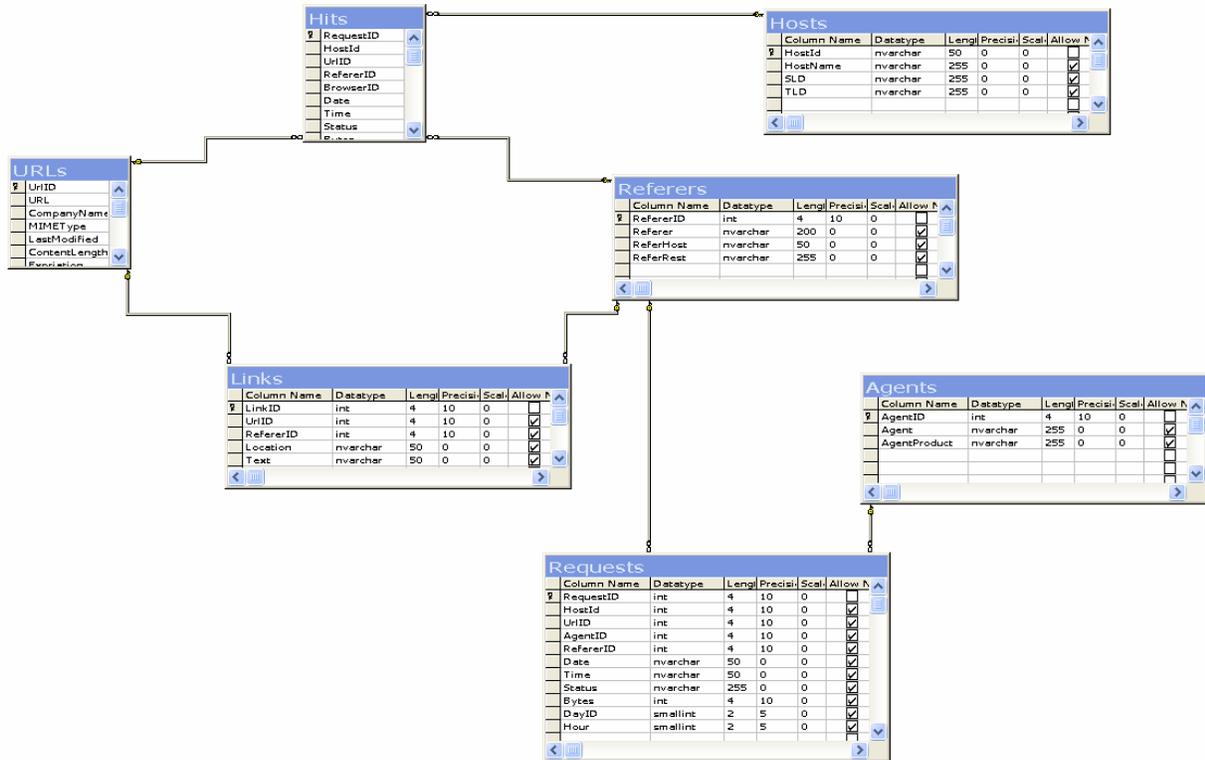


Figure 74: Database Schema showing the relationship between various tables in the weblogs database.



Figure 75: Multi-view visualization build using Fusion and visualization schema for weblogs database (Figure 74). Datafaces is shown at the left (Figure 76) and visualization workspace having 5 components is shown at the right (Figure 77).

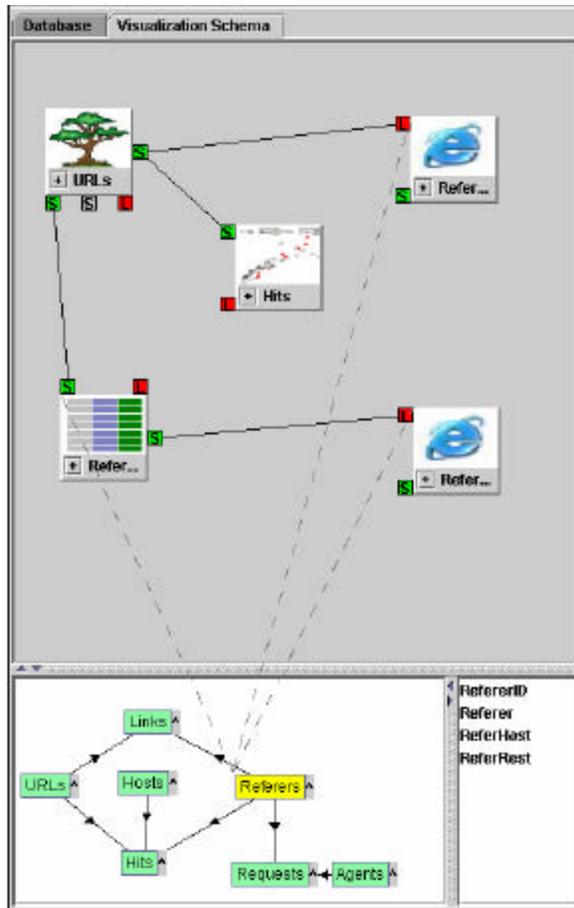


Figure 76: Datafases for the visualization constructed in Figure 75



Figure 77: Visualization workspace showing 5 coordinated components.

# Files and Folders



Figure 78: Database schema for Files and Folders. Showing a one to many relationship between folders and files.

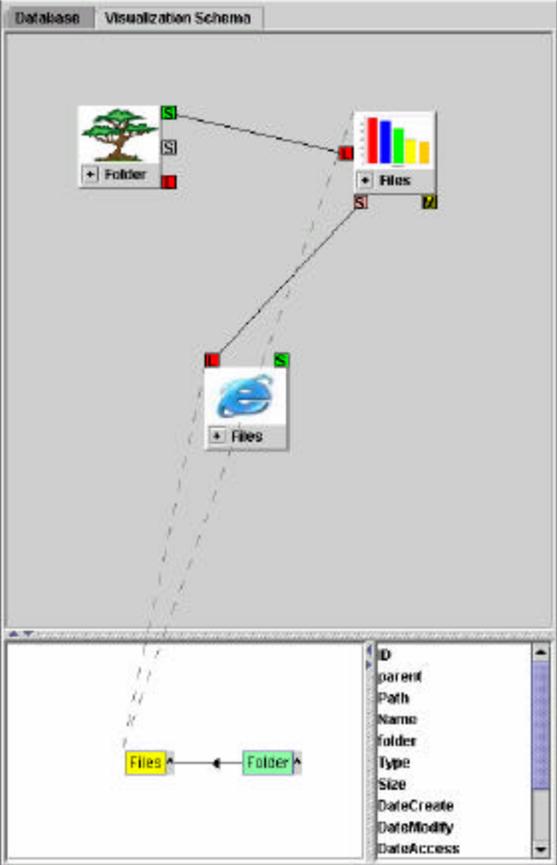


Figure 79: Datafaces showing visualization schema files above the dataschema overview.

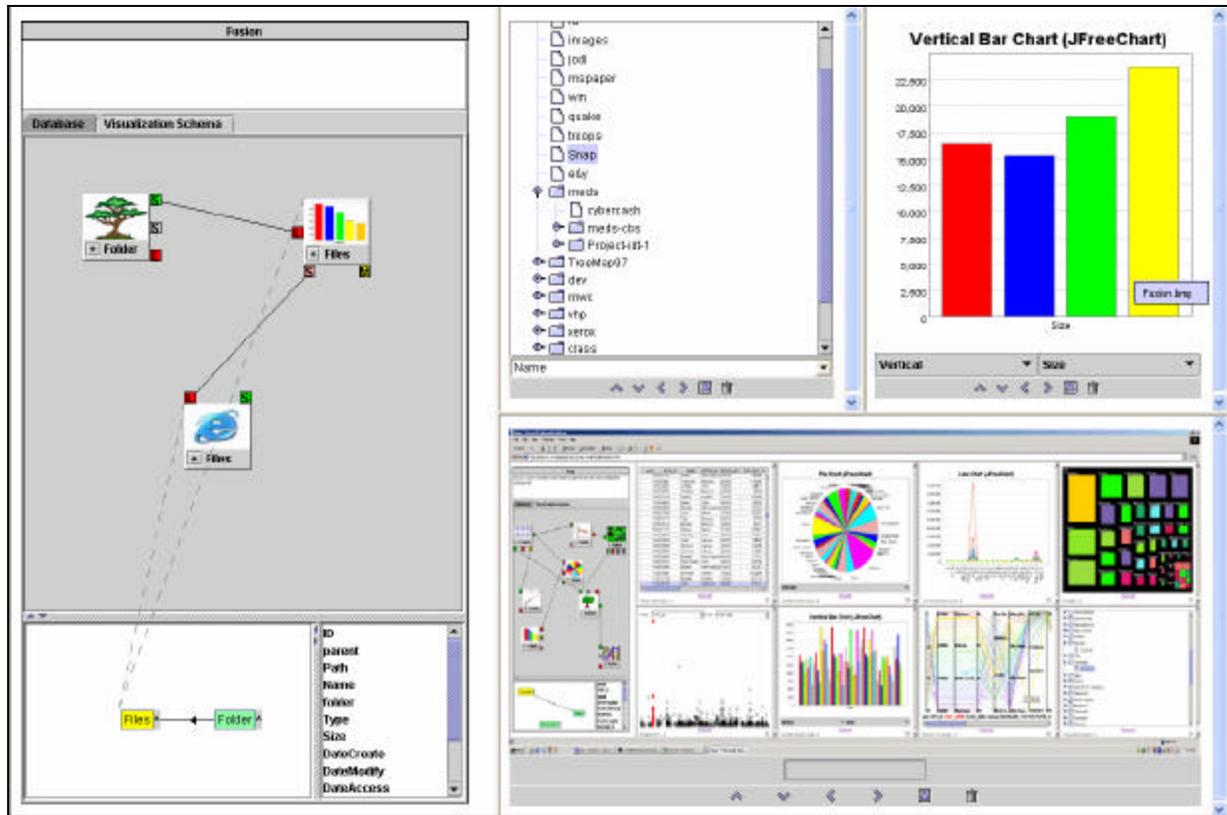


Figure 80: Fusion used as a hard disk exploration tool. The tree-view indicated the folders hierarchy. The bar chart displays the contents by size. The web page loads the contents. Above scenario displays that the largest image file in the Snap directory i.e. Fusion.bmp (loaded in bottom view)

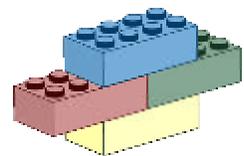
## References

1. A. Siepel , A.F., A. Tolopko , M. Zhuang , P. Mendes , W. Beavis , B. Sobral ISYS: a decentralized, component-based approach to the integration of heterogeneous bioinformatics resources. *Bioinformatics*, 17:1. 83-94.
2. Ahlberg, C., Shneiderman, B. Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. *Proc. ACM SIGCHI 1994*. 313-317.
3. Ahlberg, C., Wistrand, E. IVEE: An Information Visualization and Exploration Environment. *Proc. IEEE Information Visualization '95*. 66-73.
4. Aiken, A., Chen, J., Stonebraker, M., Woodruff, A Tioga-2: A Direct Manipulation Database Visualization Environment. *Proc. 12th International Conference on Data Engineering*. 208-217.
5. Baldonado, M., Woodruff, A., Kuchinsky, A. Guidelines for Using Multiple Views in Information Visualization. *Proc. ACM Advanced Visual Interfaces 2000*, (2000).
6. Becker, R., Cleveland, W. Brushing scatterplots. *Technometrics* 29(2). 127-142.
7. Bosch, R., Stolte, C., Tang, D., Gerth, J., Rosenblum, M., Hanrahan, P. Rivet: A Flexible Environment for Computer Systems Visualization. *ACM SIGGRAPH Computer Graphics*, 34(1).
8. Card, S., Mackinlay, J., Shneiderman, B., Readings in Information Visualization: Using Vision to Think. *Morgan Kaufmann*, (1999).
9. Conklin, N. A web-based, run-time extensible architecture for interactive visualization and exploration of diverse data. *Virginia Polytechnic Institute and State University, Department of Computer Science, Master Thesis*.
10. Derthick, M., Roth, S.F., Kolojejchick, J. Coordinating declarative queries with a direct manipulation data exploration environment". *IEEE Information Visualization*.
11. Fegaras, L. VOODOO: A visual object-oriented database language for ODMG OQL. *ECOOP Workshop on Object-Oriented Databases*. 61-72.
12. Fogg, D. Lessons from a Living in a Database Graphical Query Interface. *Proc. ACM SIGMOD'84*. 100-106.
13. Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, (1995). *Addison-Wesley*, (1995).

14. Grig-Software Compare It![WWW Document]. <http://www.grigsoft.com/wincmp.htm>, (accessed April 2003).
15. Haber, E., Ioannidis, Y., Livny, M. OPOSSUM: desk-top schema management through customizable visualization. *Proc. VLDB'95*.
16. Isakowitz, T., Stohr, E., Balasubramanian, P. RMM: a methodology for structured hypermedia design. *Communications of the ACM*, 38(8). 34-44.
17. Isenhour, P., Begole, J., Heagy, W., Shaffer, C. Sieve: A Java-Based Collaborative Visualization Environment. *IEEE Visualization 1997*. 13-16.
18. Jacqueline M. Antis, S.G.E., John D. Pyrc Visualizing the Structure of Large Relational Databases. *IEEE Software* 13(1). 72-80.
19. Larson, J. A visual approach to browsing in a database environment. *IEEE Computer*, 19(6). 62-71.
20. Lenwood S. Heath, D.H., Lucy T. Nowell, William C. Wake, Guillermo A. Averbach, Eric Labow, Scott A. Guyer, Denis J. Brueni, Robert K. France, Kaushai Dalal, and Edward A. Fox. Envision: A user-centered database of computer science literature. *Communications of the ACM*, 38(8). 52--53.
21. LiveConnect [WWW document]. <http://wp.netscape.com/eng/mozilla/3.0/handbook/javascript/livecon.htm>, (accessed November 2002).
22. Livny, M., Ramakrishnan, R., Beyer, K., Chen, G., Donjerkovic, D., Lawande, S., Myllymaki, J., Wenger, K. DEVise: integrated querying and visual exploration of large datasets. *Proc. ACM SIGMOD'97*. 301-312.
23. Mackinlay, J. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2). 111-141.
24. Mark Derthick, J.K., Steven F. Roth. An Interactive Visual Query Environment for Exploring Data. *ACM Symposium on User Interface Software and Technology*. 189-198.
25. NCGR(ISYS) [WWW Document]. <http://www.ncgr.org/isys/>, (accessed April 2003).
26. North, C. Multiple Views and Tight Coupling in Visualization: A Language, Taxonomy, and System. *Proc. CSREA CISST 2001 Workshop of Fundamental Issues in Visualization*. 626-632.

27. North, C. A User Interface for Coordinating Visualizations Based on Relational Schemata: Snap-Together Visualization. *University of Maryland, Computer Science Dept., Doctoral Dissertation, (2000).*
28. North, C., Conklin, N., Indukuri, K., **Saini, V.** Fusion: Interactive coordination of diverse data, visualizations, and mining algorithms. *ACM CHI 2003, (2003).*
29. North, C., Conklin, N., Indukuri, K., **Saini, V.** Visualization Schemas and a Web-based Architecture for Custom Multiple-View Visualization of Multiple-Table Databases. *Information Visualization, Palgrave-Macmillan,1(3), (2002).* 211-228.
30. North, C., Conklin, N., **Saini, V.** Visualization Schemas for Flexible Information Visualization. *Proc. IEEE InfoVis 2002.* 15-22.
31. North, C., Shneiderman, B. Component-Based, User-Constructed, Multiple-View Visualization. *Proc. ACM SIGCHI 2001.* 201-202.
32. North, C., Shneiderman, B. Snap-Together Visualization: A User Interface for Coordinating Visualizations via Relational Schemata. *Proc. ACM Advanced Visual Interfaces 2000, (2000).* 128-135.
33. North, C., Shneiderman, B.. Snap-Together Visualization: Can users construct and operate coordinated visualizations? *International Journal of Human-Computer Studies, (2001).* 715-739.
34. Rabenhorst, D.A. Interactive Exploration of Multidimensional Data. *Proceedings of the SPIE Symposium on Electronic Imaging 2179.* 277-286.
35. Rabenhorst, D.A. [WWW Document] Diamond.   
<http://www.research.ibm.com/people/d/Rabenhorst/gallery.html>, (accessed April 2003).
36. Roth, S., Chuah, M., Kerpedjiev, S., Kolojejchick, J., Lucas, P. Towards an Information Visualization Workspace: Combining Multiple Means of Expression. *Human-Computer Interaction Journal, 12(1&2).* 131-185.
37. SDK [WWW document] Sun's Java 2 SDK. *Standard Edition Documentation,*   
<http://java.sun.com/j2se/1.4.1/docs/index.html>, (accessed November 2002).
38. Shaw, M., Garlan, D. Software Architecture: Perspectives on an Emerging Discipline. *Prentice Hall, (1996).*
39. Shneiderman, B. Tree visualization with treemaps: a 2-d space-filling approach. *ACM Transactions on Graphics, 11(1).* 92-99.

40. Silberschatz, A., Korth, H., Sudarshan, S. Database System Concepts. *Third Edition*, McGraw-Hill, (1999).
41. Takatsuka, M.a.G., M. GeoVISTA Studio: A Codeless Visual Programming Environment For Geoscientific Data Analysis and Visualization. *Journal of Computers and Geosciences*, (2002).
42. Upson, C., Faulhauber, T., Kamins, D., Laidlaw, D., et. al. The Application Visualization System: A Computational Environment for Scientific Visualization. *IEEE Computer Graphics and Applications*, 10(4). 30-42.
43. Wang, J. VID: A Lightweight Protocol Between Visualization Systems and Digital Libraries. *Virginia Polytechnic Institute and State University, Department of Computer Science, Master Thesis*.
44. Ward, M. Creating and Manupulating N-Dimensional Brushes. *Worcester Polytechnic Institute Computer Science Department*, (1997).



## VITA

Varun is the eldest son of Kamlesh and Prakash Saini. He was born in Meerut, India in July of 1980. He earned a Bachelor of Computer Science and Engineering Degree, from the Faculty of Technology and Engineering at Maharaja Sayajirao University of Baroda in 2001.

While pursuing his Master's Degree in Computer Science at Virginia Tech, Varun worked as a Graduate Research Assistant at the AT&T Center of Scientific Visualization of Organizations.

Following graduation, Varun will be moving to Redmond, WA to pursue a career in Computer Science. This thesis completes his M.S. degree in Computer Science from Virginia Tech.

---

North, C., Conklin, Saini, V., Indukuri, K., "[Fusion: Interactive coordination of diverse data, visualizations, and mining algorithms](#)", Demonstration, *ACM CHI 2003*.

North, C., Conklin, N., and Saini, V., "[Visualization Schemas for Flexible Information Visualization](#)", *Proc. IEEE InfoVis 2002 Symposium*, (October 2002).

North, C., Conklin, N., Indukuri, K., and Saini, V., "[Visualization Schemas and a Web-based Architecture for Custom Multiple-View Visualization of Multiple-Table Databases](#)", *Information Visualization*, Palgrave-Macmillan, (December 2002).