CHAPTER 2

LITERATURE REVIEW

In this chapter, we develop our definition of database exploration, and also review related research concerning user modeling and numerous aspects of database-visualization integration. Related research in database exploration has both *user-centric* and *data-centric* aspects. User-centric aspects relate to previous attempts in modeling users as data explorers and monitoring users while interacting with data. Data-centric aspects relate to previous attempts in developing scientific database systems, integrating databases with visualization, and constructing knowledge structures (metadata) to model components of the exploration. This review is especially broad, because we are looking at a process that has many components.

2.1 Definition of Database Exploration

Our definition of database exploration builds upon the traditional definition of exploratory data analysis, with an emphasis on visualization and database components. Database exploration is an inherently interactive activity. It utilizes numerous data representations, such as textual reports, graphical plots and numerical summaries. It also exhibits numerous data interactions, such as database queries, analytic computations, navigations through data spaces and recording of insights. A fundamental result of this

process is the evolution of hypotheses about the data and the generation of models that accurately describe the data under investigation.

2.1.1 Statistical Analysis Origins

Database exploration has origins in statistical data analysis and is almost synonymous with data exploration; the key difference is that the data resides in a database, and is managed by a database management system (DBMS). Data exploration is an alternate term for *exploratory data analysis* (EDA), a phrase coined by Tukey, who offered the following definition:

> "Exploratory data analysis is detective work - numerical detective work - or counting detective work - or graphical detective work ... unless exploratory data analysis uncovers indications, usually quantitative ones, there is likely to be nothing for confirmatory data analysis to consider ... [it] can never be the whole story, but nothing else can serve as the foundation stone - as the first step (Tukey 1977, p1-3)."

Tukey's definition contains two important elements. The first is that EDA takes on many forms (numerical, graphical, etc.). The second is that EDA is often the first stage of a data analysis, serving to highlight promising areas within a database for in-depth investigation. Velleman (1990) characterizes EDA as the process of seeking patterns, relationships and anomalies from data, with an emphasis on data manipulations. The data exploration model of Jambu (1991) outlines distinct stages of the EDA process, from simple analyses of each distinct data variable, to dependency and relationship analyses among variables, to advanced factor analyses using geometric data encodings. Confirmatory data analysis (CDA) is subsequently used to verify data distribution assumptions, fit models to summarize data structures and quantify random error (Nicholson 1983). Velleman's characterization concludes that the primary software challenge is to integrate EDA techniques, graphics and calculations with CDA methods.

2.1.2 Visualization Considerations

Visualization techniques are particularly useful for data exploration, because they can harness human perceptual capabilities in comprehending massive amounts of data. Visualization also allows greater interactions with data because of the additional input devices, user interface toolkits and direct manipulation techniques (Schneiderman 1983) that allow the user to control many aspects of a data exploration. Thus, visualization is an input as well as an output technology that can be used for data exploration.

Visual exploratory data analysis (VEDA), as defined in Young, Kent, and Kuhfeld (1988), shifts the focus from traditional EDA towards describing data visually and being primarily multivariate. Static VEDA methods include organizing matrices of 2D scatterplots, or parallel coordinate displays, first described by Inselberg (1985). Scatterplots aid in the discrimination of strata, or multiple groups, within a bivariate data display (Spense and Lewandowsky 1990). Scatterplots can also be extended with iconographic representations (Grinstein, Pickett and Williams 1989; Grinstein, Sieg, Smith and Williams 1992; Ward 1994) to increase the dimensionality of data displayed. Dynamic VEDA methods include linked displays that can be manually probed via brushing (Becker and Cleveland 1987; Buja, McDonald, Michalak and Stuetzle 1991; Ward and Martin 1995), or data projections that are algorithmically computed such as plot interpolations, correlation tours and grand tours (Buja, Asimov, Hurley and McDonald 1988).

2.1.3 Database Considerations

Database management systems are essential for data exploration, because they enable data selection through powerful, standardized query facilities, and manage persistent data objects. DBMSs provide both the structured data representation and means

for accessing data based on relationships and associations. This is crucial for intelligent data reduction. Also, since there are tremendous amounts of data stored in corporate relational databases, it only makes sense that databases be considered central to any exploration environment.

The field of Knowledge Discovery in Databases (KDD) has generated a great deal of interest recently (Piatetsky-Shapiro and Frawley 1991, Fayyad, Piatetsky-Shapiro, Smyth and Uthurusamy 1996). The fundamental goal of KDD, the nontrivial extraction of structure and patterns from databases, is very similar to data exploration. KDD research primarily focuses on automated data analysis, using decision trees, linear and nonlinear regression, classification, nearest-neighbor techniques, and pattern-matching methods (Fayyad, Piatetsky-Shapiro and Smyth 1996).

2.1.4 Interaction Considerations

The dictionary definition of interaction is “a mutual or reciprocal action or influence”. It implies two cooperating participants; in our case, they are the human data explorer and the software application that manipulates data, as shown in Figure 2-1. The user conceptually interacts with the data, but this interaction is mediated by the database and visualization systems. Thus, there are both conceptual user-data interactions and supportive system-level interactions. Such interactions have been mentioned recently (Velleman 1990; Brachman and Anand 1996; Foley 1995; Thomas 1995; Keim 1996).

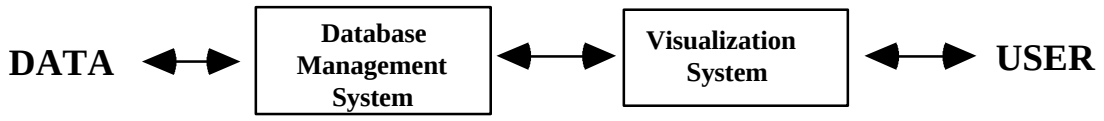


Figure 2-1. User-Data Interaction Through Visualization and Database Systems.

2.1.5 Database Exploration

We define database exploration as *the process of extracting knowledge from databases using visual, analytic and database tools*. This definition is essentially no different than Tukey's. It, however, emphasizes the fact that it is a *process*, an interactive activity that is undertaken by both the user and computer. It also emphasizes the visual component of knowledge extraction, because it is through visualization that the user orchestrates all activities, perceives data configurations, composes analyses and produces results. Our definition focuses on the manual search for knowledge, with support from automated tools.

2.2 User-Centric Aspects of Database Exploration

There have been a number of descriptions of the general properties of the database exploration process. Young and Smith (1991) described an exploratory "cognitive mode" of data analysis in terms of *goals* (externalizing ideas, considering possibilities), *constraints* (few or loosened to encourage creativity), *processes* (memory recall and associative thinking) and *products* (concrete representations of ideas, clusters of related concepts). Khoshafian, Bates and DeWitt (1985) described the database exploration process in terms of browsing, sampling, testing and editing data subsets against hypotheses. Velleman (1990) identified four types of exploratory methods: *revelation*

(revealing patterns in data), *residuals* (removing a fitted model to the data), *re-expression* (simplifying the analysis by transforming the data) and *resistance* (protection against outliers or bad data). O'Day and Jeffries noted that virtually all planned and exploratory searches involve several steps, have winding paths and an occasional unexpected turn. Common progressive searching involves an *orienting* overview followed by a detailed *search* based on the orienting result.

Since our research focuses on the interactive nature of database exploration, we need an understanding of how users interact with data. We borrow some of the human-computer interaction (HCI) community terminology for our basic definitions. Specifically, we utilize the terminology of Card, Moran and Newell (1983) to define goals, tasks, plans and actions. *Goals* describe the desired result the user wishes to achieve; *tasks* are the activities believed by the user to achieve the goals; *plans* are implementation of the tasks; *actions* are simple tasks that involve no problem solving or control structure component (such as input device manipulation). Plans are often linked sequences of actions. The database exploration system responses are then fed back to influence the tasks, plans and goals.

The terminology of Card, Moran and Newell, (GOMS: Goals, Operators, Methods and Selection Rules), is similar to ours. We, however, use the term *plans* instead of methods and *actions* instead of operators. Our purpose is also slightly different. Our goal is to describe the user-data interactions of database exploration. The GOMS goal is to enable time measurements of user tasks, for performance and user interface enhancement.

In applying our framework to describing user-data interactions, we consider interactions to be sequences of interconnected actions the user executes, *along with the system responses*, as shown in Figure 2-2.

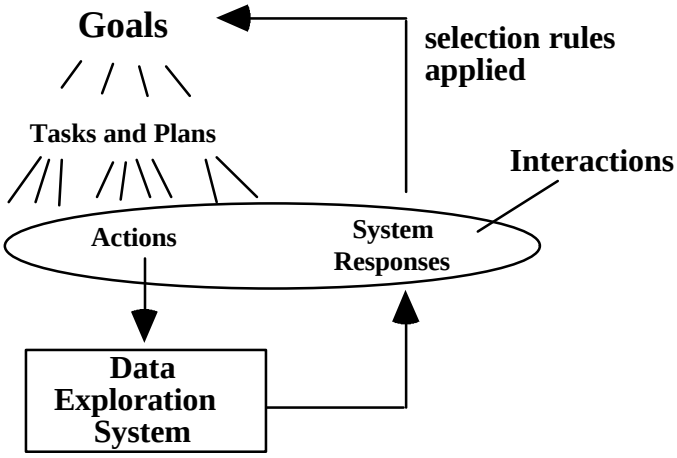


Figure 2-2. Data exploration interaction framework.

The user goal is to explore the data in order to become familiar with it, and ultimately draw some conclusions about it. Between familiarization and conclusions, there are numerous interactions that can occur to uncover meaning within the data. As described in Norman (1986), interactions require actions to be undertaken, and their results evaluated against the goals for the exploration to proceed. Based on the perception and evaluation of the system responses, selection rules are applied to choose the next task, plan or action.

The interactions then become a database by which the user can be analyzed and modeled as a data explorer. Interactions are not well-defined, or predetermined for database exploration, so we first focus on the actions that the user performs to gain insight into the database exploration process. We partition the definition of a database exploration interaction model through actions into

- A *unit-task-level* description of the fundamental database exploration elements. This forms the vocabulary of database exploration interactions.

- A *process-level* description of how the fundamental task elements are structured, sequenced and interrelated to form higher-level *interaction constructs*. This forms the kernel of a database exploration interaction language.

2.2.1 Task-Level Descriptions of User-Data Interaction

There have been numerous attempts at identifying and describing the tasks that make up database exploration, ranging from low-level taxonomies to very general summarizations. A useful task taxonomy is offered by Springmeyer (1992). This empirical study of domain scientists analyzing data contains the most elemental of tasks, shown in Figure 2-3. The model contains two high-level categories, *Investigation* and *Integration of Insight*, with four mid-level subcategories *Interacting with Representations*, *Applying Mathematics*, *Maneuvering* and *Expressing Ideas*, and numerous low-level task types. These were determined using an interaction analysis, disregarding the possible effect of the tool being used.

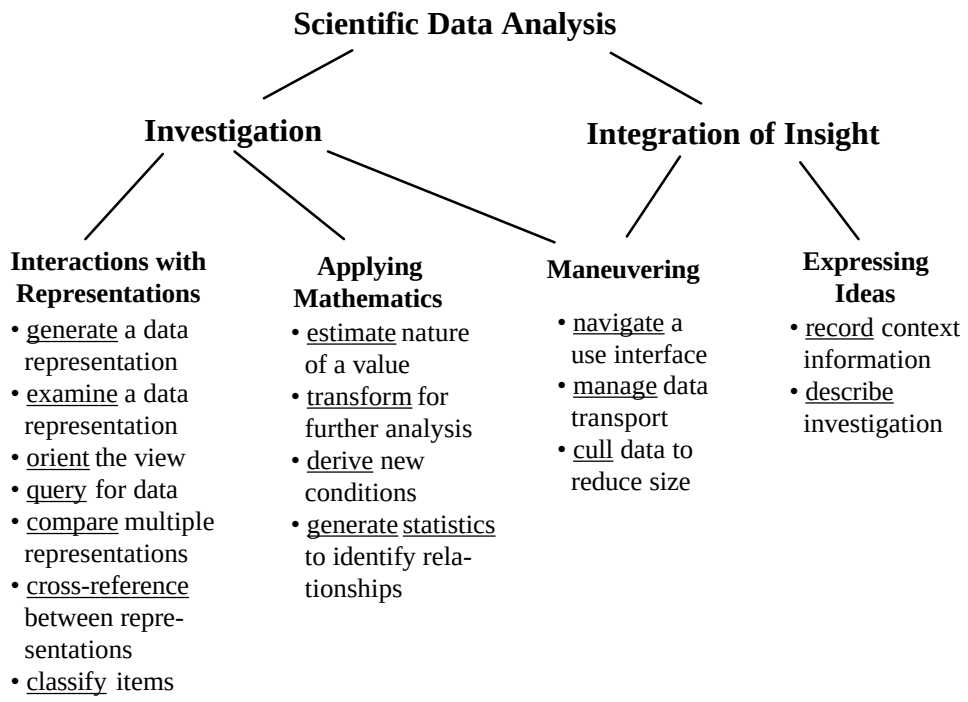


Figure 2-3. Springmeyer's task taxonomy (from Springmeyer 1992).

We partition prior database exploration task research into two broad categories, based on the domain perspective taken: visualization or database, and compare each with Springmeyer's taxonomy, to observe the commonalities. It is evident from this review that Springmeyer's model is a superset of all other models.

2.2.1.1 Visualization Perspectives

Two recent task analysis studies of visualization interactions in different domains described similar fundamental tasks. Knapp (1994) analyzed geographers using a visualization software application, and identified the following primary visual tasks: *identify, locate, compare* and *associate*. These embody a logical progression of a typical analysis of geographical data. Secondary subtasks identified included *distinguish, categorize, cluster, rank, distribution* and *correlate*. Knapp's definitions are primarily

higher level than Springmeyer's, involving perception, evaluation and recognition. For example, *identify* requires examine, query and cross-reference tasks. Most of the secondary subtasks require examinations and comparisons to classify and order data representations.

Gillan (1993) analyzed humans interacting with simple graphic displays, and identified *searching* for a spatial location of a variable indicator on a graph, *encoding* a value of an indicator, performing *arithmetic* operations on encoded values, *comparing* spatial relations and *responding*. It was observed that computational processes were used, and their ordering depended on the graph in addition to the question to be answered (goal). Encoding a value is equivalent to generating a representation. These tasks all exist in Springmeyer's taxonomy.

Wickens (1993) considered cognitive elements in his assessment of visualization tasks. Because of the subjective nature of visualization, tasks possess varying levels of uncertainty, based on user confidence of the investigation or visualization. He identified three major tasks: *searching, comparing* and *understanding*, each having a correlate in Springmeyer's taxonomy. This subjective nature was also expressed in Buja, Cook and Swayne (1996), though not closely tied to cognition. Three basic interactive data visualization tasks were described in the context of a multidimensional graphical data analysis system: *finding Gestalt, posing queries* and *making comparisons*. Finding Gestalt incorporates the user perceiving and evaluating a visualization to find structure.

The domain of automatic visualization strives to create accurate visualizations based on knowledge of the data and the goal to be satisfied, relieving the data analyst of many burdens associated with perception, graphics, user interfaces, etc. In most cases, there is some notion of a task to be completed as an input parameter. Wehrend and Lewis

(1990) cataloged data visualization techniques based on an objects vs. operations matrix. Perceptual tasks include *compare, categorize, identify, locate, distribution, rank, associate, locate, distinguish,* and *correlate*. Casner (1991) identified task-specific ways in which graphs are useful, making *searching* and *computation* easier. Logical tasks were specified as functions of logical operations consisting of database queries and computations over query results. Perceptual tasks are search operations to locate graphical objects or attributes, and computation operations to perform discriminations among graphical properties.

Ignatius and Senay (1992) described two basic tasks, *search* and *comparison*, as visual and cognitive operations on data subsets. These are modeled after Casner's basic task operators. They also added two other computation tasks: *correlations* and *comparisons*. Robertson (1991) described user interpretation goals as *discriminations* of graphical attributes of data distributions. Isolation methods for subsets of attributes that describe general data characteristics include values at a point, gradients showing local distributions of values (features), and trends showing global distributions of values (structures). Beshers and Feiner (1993) use a rule-based language to support visualization task operators *explore*, *directed search* and *compare*, specified on a set of relations.

2.2.1.2 Database Perspectives

Database systems research focuses much energy on data representations and query processing to provide adequate data storage and selection for a variety of application domains. Some recent research has explored supporting computations over scientific data (Wolniewicz and Graefe 1993). Querying is a means of navigating through a database. Larson (1986) identified four components of the *browsing* query operation that were visual in nature: *structuring* data for output, *filtering* data, *panning* and *zooming*. He

describes some graphical user interface mechanisms for specifying these operations for accessing relational data.

The empirical analysis of O'Day and Jeffries (1993) studied the effectiveness of information retrieval tasks. Six types of analysis were noted: *identifying* trends and correlations, making *comparisons*, *experimenting* with different aggregates and/or scaling, *identifying* critical subsets or unique items, making *assessments*, and *interpreting* data to define terms and concepts. Brachman et al. (1993) identified analysis tasks and sub-tasks as *segmenting* classes with different attributes and parameters, *viewing* and *comparing* data, and *saving* and *re-using* work (both analysis results and techniques). Brachman and Anand (1996) defined three basic knowledge discovery tasks: *queries* to the database, *analysis* and *visualization* of data sets (a component of insight), and *presentation* of output.

2.2.1.3 Task-Level Interaction Summary

Both low-level and high-level task taxonomies exist in the literature. The high-level taxonomies can be decomposed into low-level tasks in a number of different ways, depending on the user and application. High-level constructs, such as *identify*, can require a query, a visual examination, or a classification, for example. High-level constructs might also require several low-level tasks to complete, such as correlation, which would involve a comparison and a generation of some statistic to describe the relationship. Also, correlation might not result in a numeric quantity generated, but a general assessment of the relationship. With such differences noted, we present our comparison of the prior task research against Springmeyer's base level task taxonomy in Table 2-1.

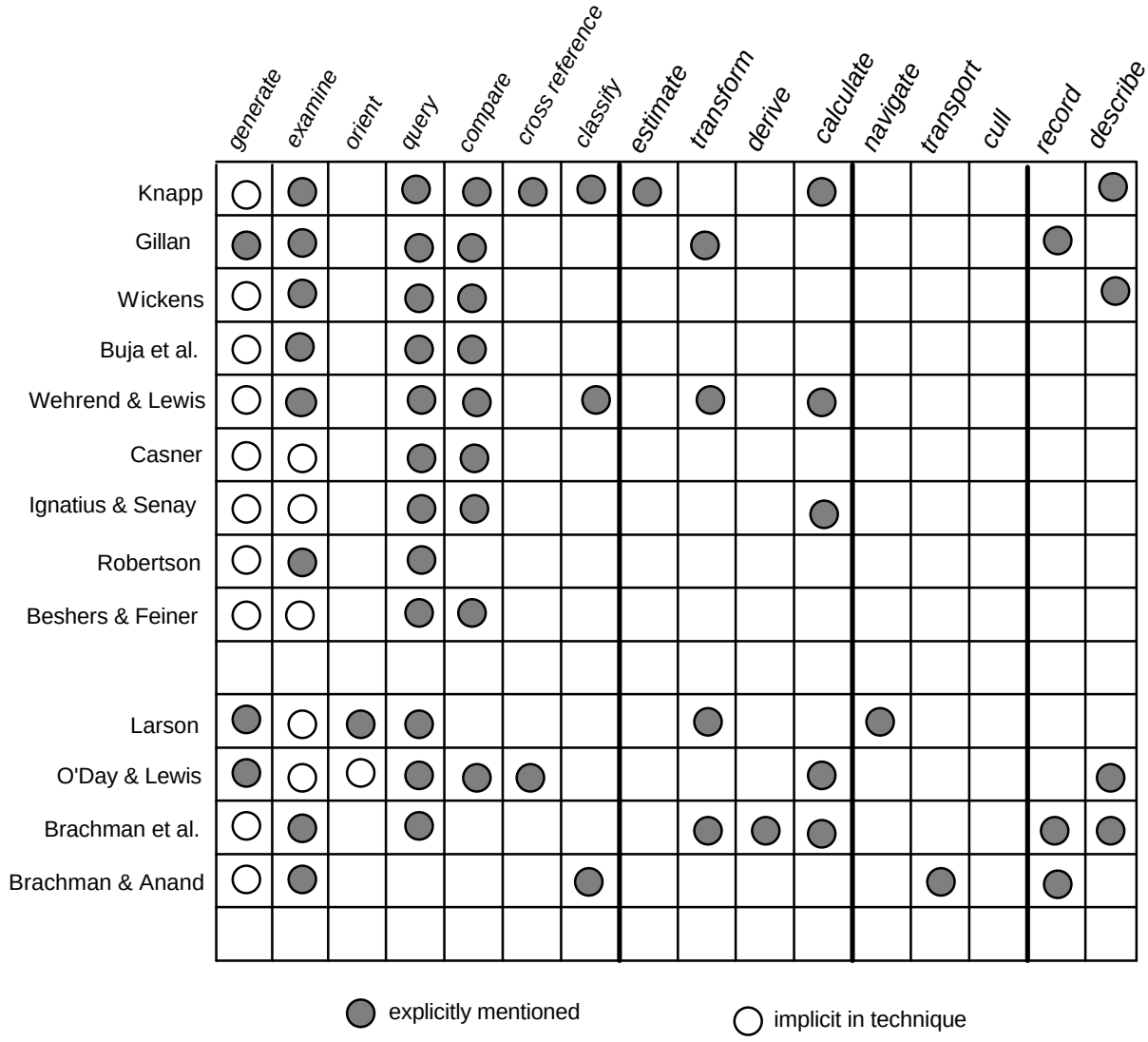


Table 2-1. Comparison of task-level user-data interactions.

The *generate* and *examine* tasks are universal throughout, though they are not explicitly mentioned in every study, as data representations must be created and evaluated in any data analysis. Querying and comparing are the most frequently mentioned tasks. Also, the database approaches seem to focus on other issues aside from interacting with representations, devoting more effort towards data management, recording and analysis.

In summary, the studies of Springmeyer and Knapp are more complete in terms of coverage of different types of tasks, most likely because they are in-depth interaction analyses. The other studies devote more effort to a smaller set of tasks in specific domains.

2.2.2 Process-Level Descriptions of User-Data Interaction

Tasks do not exist in isolation. In between each task activation is an evaluation process undertaken by the user, as shown in Figure 2-2. The system response to each user action must be perceived, interpreted and evaluated against the current goal before the next task is formulated, as detailed in (Norman 1986) and diagrammed in Figure 2-4.

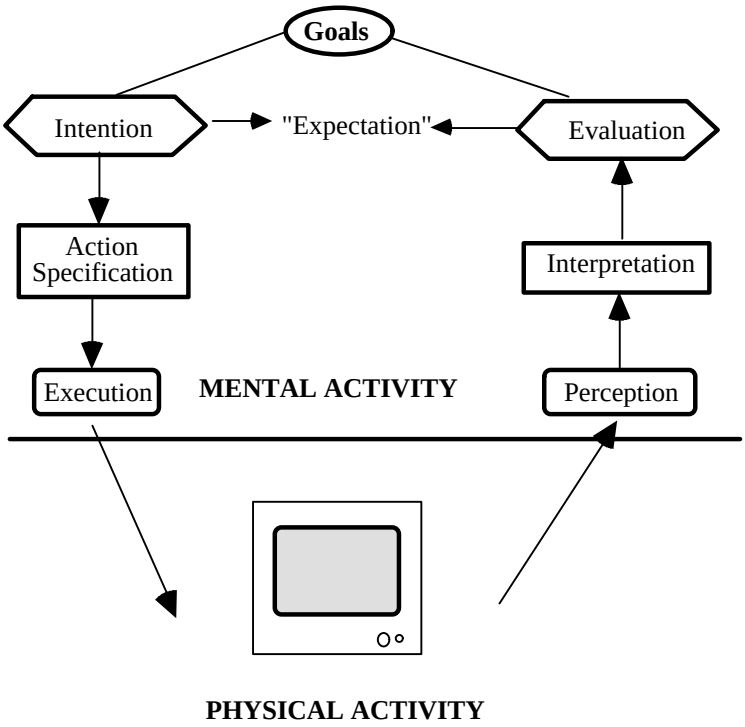


Figure 2-4. The stages of user activity in the performance of a task (from Norman 1986).

With respect to database exploration, the system is an "active participant" by providing answers to questions not completely formulated, as postulated by Owen (1986). Velleman offered this elegant and concise description:

> "Data exploration follows threads of investigations in which each display or analysis suggests something of interest. The analyst doesn't know what he will choose to do next until he sees the results of what he has just done."

In turn, this progression of interactions builds a "rich mental environment", a context that embodies the user's mental model of the database and the exploration state attained (Cypher 1986). Process descriptions are often graphical, to show structures and relationships more clearly than textual or symbolic representations. We now look at process models that include multiple tasks, and use graphical representations.

2.2.2.1 The Human Problem Solving Process

An analysis of human problem solving in various task environments described in Newell and Simon (1972) captures the behavior of a user traversing a problem space in a *problem behavior graph*. The problem space consists of a set of knowledge states, operators for changing between knowledge states, constraints for applying operators and control knowledge for deciding which operators to apply. Note that some of these concepts evolved into the GOMS model described in Section 2.2.

In the problem behavior graph, vertices represents states of knowledge and horizontal directed edges represented the application of an operator. Undirected vertical edges between vertices represent backtracking and the addition of duplicated vertices. Repeated operator application create multi-lined edges. Figure 2-5 shows the primary problem behavior graph components.

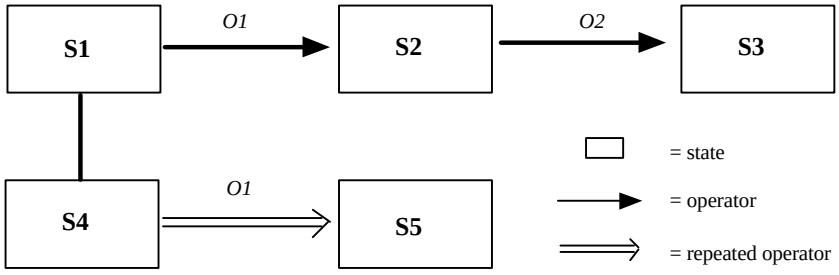


Figure 2-5. Components of a *problem behavior graph* (from Newell and Simon (1972), p. 173).

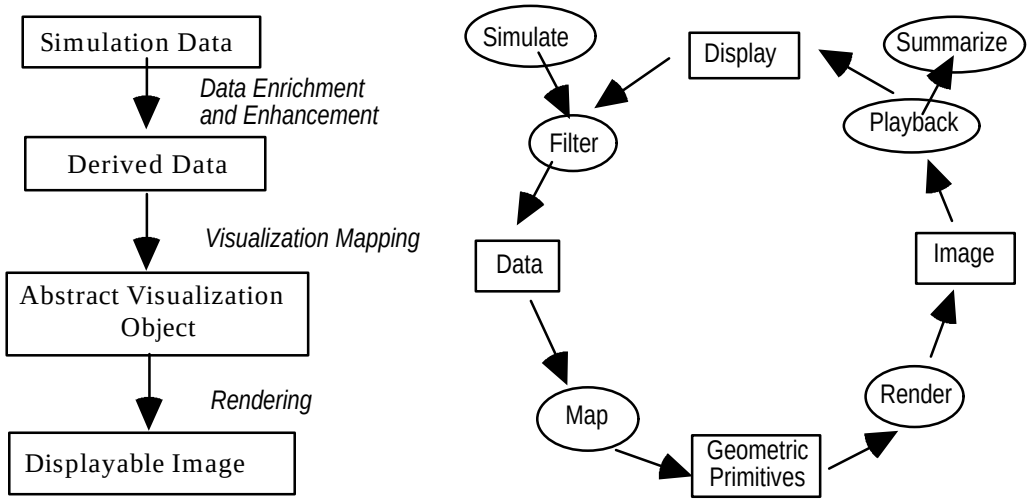
Newell and Simon conducted interaction analyses of subjects performing specific tasks: solving a cryptarithmetic puzzle, a logic expression, and a chess problem. Two graphical representations were created, one a large, detailed version that contains all the information gathered, and the other a compact, summary version, shown in Figure 2-6.

It is the summary version that is more interesting and informative, since the graphs created can get very large (a vertex added for each action taken by the subject). The summary graph shows gross processes much more clearly, as Figure 2-6 shows for the cryptarithmetic problem. The graph grows in time from left to right, and from top to bottom. The notes on the right describe the major tasks undertaken and conclusions drawn.

Figure 2-6. The problem behavior graph for the cryptarithmetic problem of Newell and Simon (1973).

2.2.2.2 Visualization Process Models

Haber and McNabb (1988) introduced the visualization pipeline model, shown in Figure 2-7a. This model involves three tasks: 1) Raw simulation data is *filtered* (normalized, subsampled, interpolated, enhanced, etc.) into some derived data; 2) derived data undergoes a *visualization mapping* into some abstract or logical representational entity, where data values are used to alter the visual attributes (such as geometry or color) of the abstract entity; 3) the final *rendering* transformation to the display representation incorporates aspects of viewing, drawing and shading. Upson et al. (1989) modified the visualization pipeline to become a cycle, to emphasize the cyclic, output-as-input nature of scientific investigation, as shown in Figure 2-7b.



a. pipeline (Haber and McNabb 1988)

b. cycle (Upson et al. 1989)

Figure 2-7. The visualization pipeline and cycle models.

Bergeron and Grinstein (1993) identified 4 basic visualization interactions: 1) modifying the visualization process, 2) modifying visualization transform parameters, 3)

interaction with the visualization output to change the process, and 4) interaction with the data being processed. These were described as interactions, not tasks, so speak more to the process of database exploration composed of tasks. Felger, Fruhauf, Gobel, Gnatz, and Huffman (1991) envisioned a basic model for visualization systems represented as a bipartite graph containing *storage* nodes and *process* nodes that represent data sources (input devices and sensors), data absorbers (displays and plotters) and data transformers (sampling, enrichment, enhancement, etc.). This is similar to the Haber model, but contains a finer resolution of distinct process and storage nodes. A refinement in Felger and Astheimer (1991) emphasizes two types of user interaction: interaction with data and interaction with configuration and control, as shown in Figure 2-8. Interacting with configuration and control is similar to the visualization process interactions described in Bergeron and Grinstein (1993). Note that data flows between the data sources and the display in either direction, and the user can interact with any stage of the visualization process, or the data at the display itself.

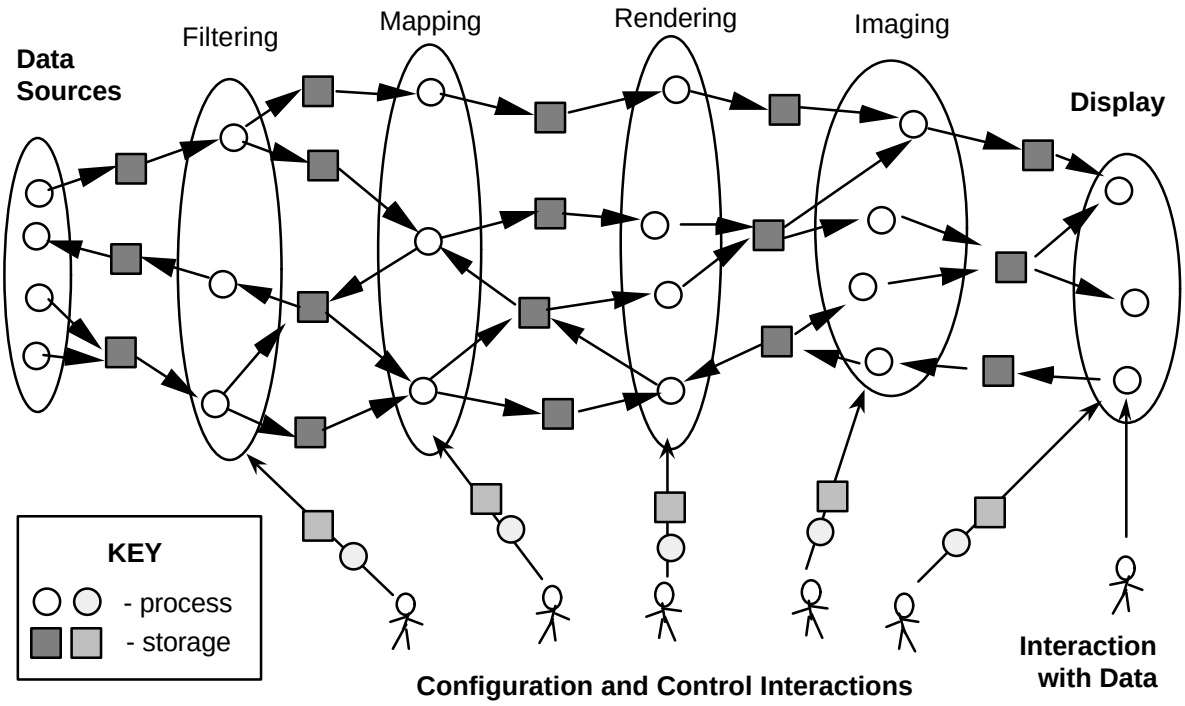


Figure 2-8 Scientific visualization process network of Felger and Astheimer (1991).

Brodlie, Poon, Wright, Brankin, Baneck and Gay (1993) described a system for monitoring computational simulations, comparing results of different simulation runs, and restarting previous computations with new parameters. They devised a dynamic data model to accommodate these operations. A *history tree* structure (see Figure 2-9) is created to store the partial simulation results, which are written to a log at adjustable time intervals. History Tree records contain the timestamp, simulation expression parameters and potentially the data being investigated in an application-specific format. The analyst could interact directly with this tree to select a saved state point, change parameters, and restart a new branch of the simulation.

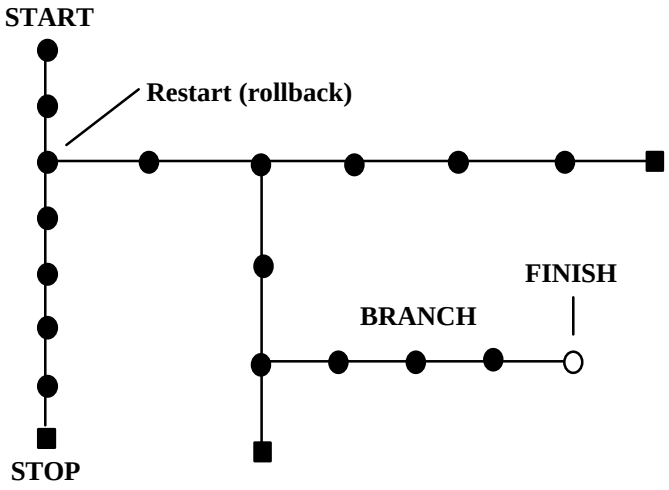


Figure 2-9. The history tree process model of Brodlie et al. (1993).

2.2.2.3 Database Process Models

Canter, Rivers and Storrs (1985) use graphical methods to portray network database navigation, and record the interaction "patterns" for an information retrieval application. Six graphical indices characterize navigation: pathiness, loopiness, ringiness, spikiness, ratio of total nodes visited versus total nodes and ratio of total distinct nodes visited versus total nodes. These indices are assigned measures to define five navigation "strategies": scanning, browsing, searching, exploring and wandering. Figure 2-10 shows the *scanning* strategy. This model might be useful in describing World Wide Web navigations.

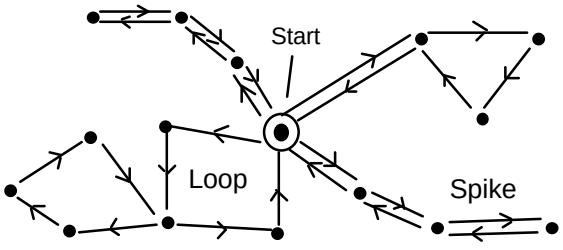


Figure 2-10 Scanning search strategy of Canter et al. (1985).

Hachem, Serrao, Gennert and Qiu (1994) proposed a Petri-Net model for managing data and metadata derivation histories for scientific data. Derivations are analytic processes, and are captured in a *process* object, a mapping definition between a set of input data classes and a single output class. Derivation semantics are defined beforehand among domain data classes called *concepts*. Figure 2-11 shows a typical data derivation, an unsupervised classification analysis performed on a Landsat image to derive land use / land cover.

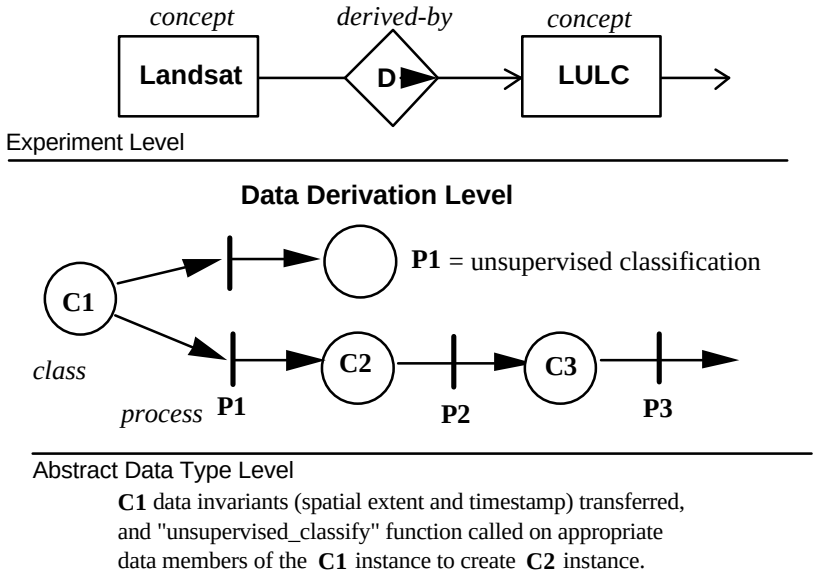


Figure 2-11 The data derivation model of Hachem et al. (1994).

Kersten and de Boer (1994) developed a model of the database browsing session as a sequence of interrelated, overlapping queries. Their framework is based on a *query dependency graph* representation that describes the search space and the actions taken by the browsing session optimizer. The browsing model defines *convergent* sessions as having consecutively more restrictive predicates, *random* sessions as having no query overlap, and *coherent* sessions as having partial query overlap, as shown in Figure 2-12.

Graph nodes contain the queries issued, and edges are labeled with the selection predicate and selection list to identify the partially re-used query results.

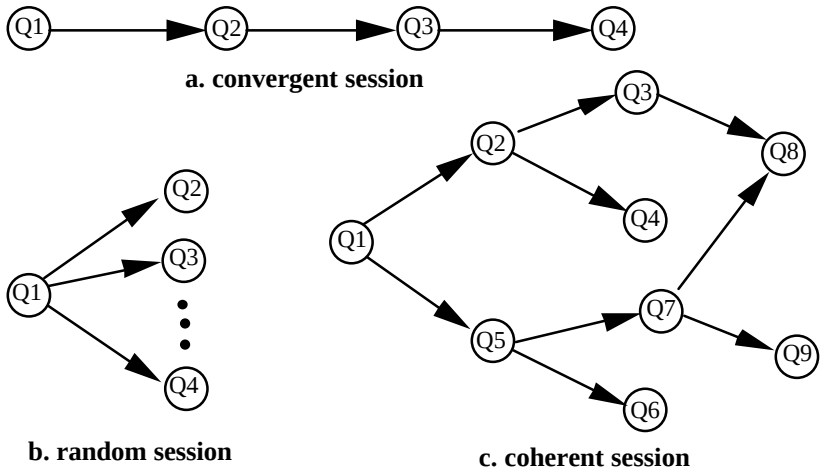


Figure 2-12. Browsing session model of Kersten and de Boer (1994).

Chuieh and Katz (1994) modeled the Computer Aided Design (CAD) process to track design data evolutions. They developed a *design thread* representation consisting of an augmented derivation graph showing use- and used-by relationships of the different versions of CAD design objects. The design thread contains a branching sequence of *history records*, that represent task invocations, as shown in Figure 2-13. Each edge of the graph represents a CAD tool invocation. A single update semantics approach is taken, where new object versions are added to the graph, as opposed to overwriting an existing, older version. Each time a new object version is created, the system automatically extracts its version, equivalence and configuration relationships from the graph.

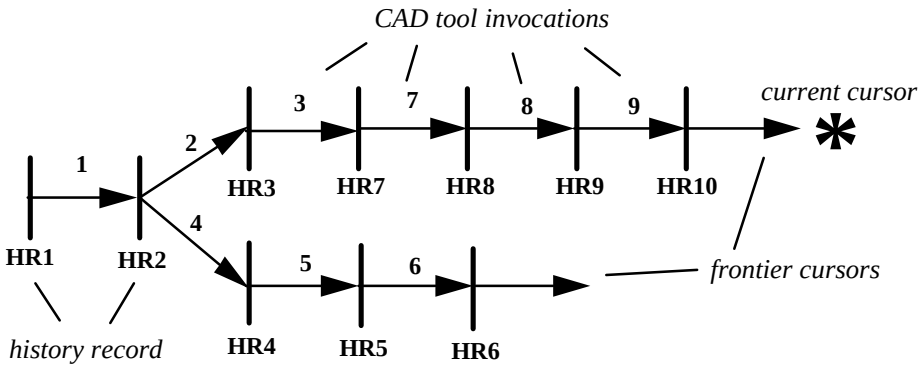


Figure 2-13. The history design thread of Chiueh and Katz (1994).

Other database approaches, targeted at the World Wide Web include analyzing browsing strategies (Catledge and Pitkow, 1995) and supporting navigation via history mechanisms (Tauscher, 1996). Catledge and Pitkow developed a relationship between the length of a path (sequence of linked web pages) and its frequency of occurrence during their study. This metric is then used to characterize web users as *serendipitous* browsers, *general purpose* browsers and *searchers*. A frequent navigational pattern within single web sites noted was the *spoke-and-hub* pattern, where central (index) pages are visited frequently, in between pages linked to them. The work of Tauscher generated single-valued metrics such as recurrence rate, page vocabulary frequency of visits, page locality and longest repeated subsequence to gain insight into the uses of history mechanisms.

2.2.2.4 Data Analysis Process Models

Becker and Chambers (1988) introduced the concepts of an *audit file* and associated *audit plot* for capturing the linear sequence of data analysis steps. The audit file contains the linear sequence of analysis commands executed, and provides a facility for investigating dependencies among analysis steps. A separate program can read the audit file and create the audit plot or answer simple queries, such as showing how an object was

generated, showing all statements contributing to a plot or result, finding all references to an object and producing scripts that could reproduce certain points of an analysis. The audit plot consists of a directed acyclic graph whose nodes are arranged in a circle. Edges connecting statements are based on the object previously assigned.

Nicholson, Carr, Crowley and Whiting (1984) defined a data analysis management system based on a network of analysis paths consisting of linked data "state" nodes. Links between states are created manually by the analyst. This is geared towards managing the entire data analysis process: reviewing the process, verifying work completed and continuing prior analyses. The realization of this system in Carr, Nicholson and Crowley (1986), focuses on data access and documentation issues. A tree structure was developed to show the data subsets with their descriptions, as shown in Figure 2-14.

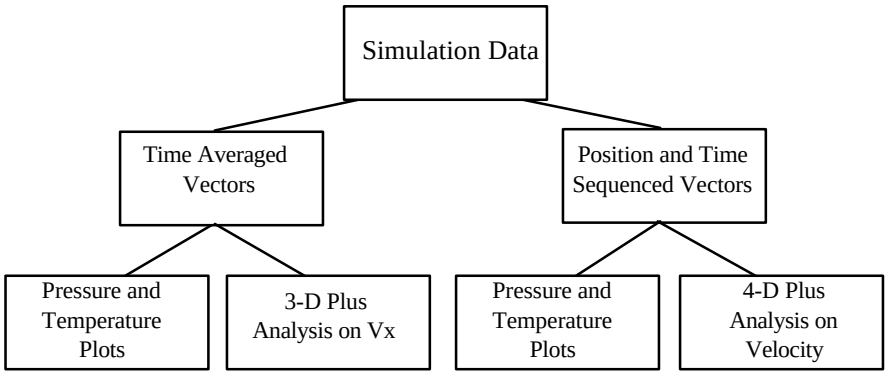


Figure 2-14. A data state tree from Carr et al. (1986).

Oldford and Peters (1986, 1988) prototyped a system that would both track and analyze data analyses, using a network data model. Nodes representing analytic objects contains various links to other objects based on causal and analytic relationships. Links are labeled with the analytic operation performed. A logical representation is continually and automatically updated with each interaction. A physical representation, the *AnalysisMap*,

shows the analysis session, which can differ from the logical representation, due to filtering, compression, aggregation of nodes, etc. Several types of views of the logical network are available: *AnalysisMap* (the logical session flow), *CausalMap* (causal relationships between objects), *DataFlowMap* (identify required variable lineage), *MicroscopicView* (in-depth data) and *ToolBox* (set of analysis step templates for use). Figure 2-15 shows an *AnalysisMap*.

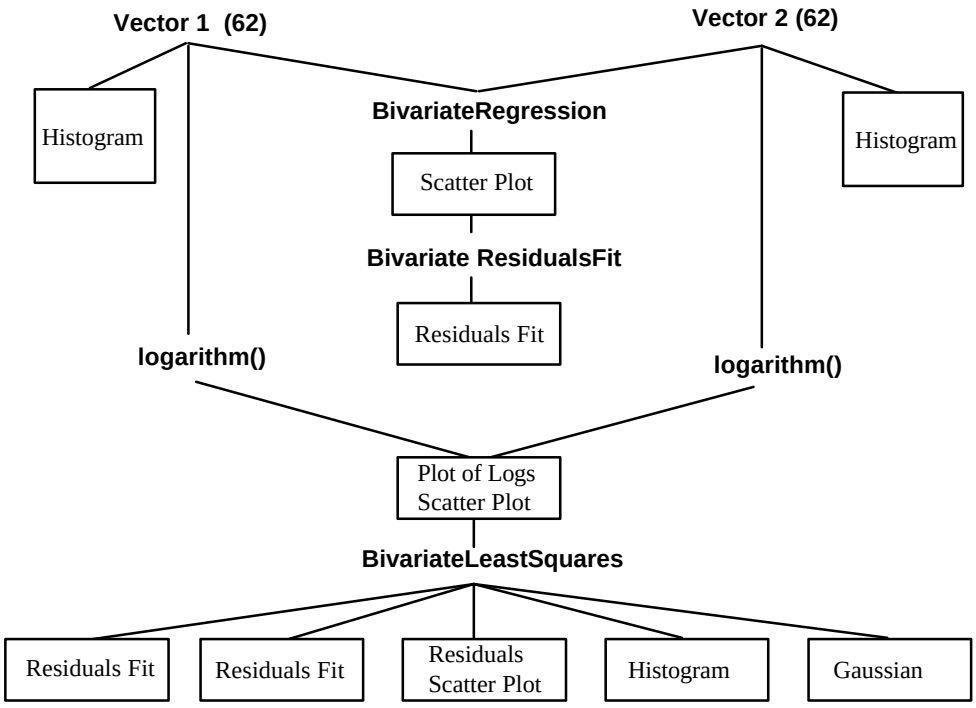


Figure 2-15. An AnalysisMap, adapted from Oldford and Peters (1986).

Young and Lubinsky (1995) also represent analyses as graphs, and apply the creation of a "statistical strategy" by an expert to assist the naive data analyst in the task of exploring data. A statistical strategy is a preconfigured analysis graph for a particular data set, called a *GuideMap*. A data analyst can use this static strategy, or deviate from it, and all the while the system automatically updates a unique and dynamic *WorkMap* analysis

graph. Thus, by using a predetermined analytic process, a data analyst can gain from an expert's approach to analyzing an unfamiliar data set. Figure 2-16 shows two examples of guidemaps, one for the entire analysis cycle, and the other for the *explore data* process. It also shows a user workmap. Graph nodes are actually buttons in the implementation, that can be highlighted or activated. Possible activations include displaying hypertext describing the button action, and initiating hypercode to realize the button action.

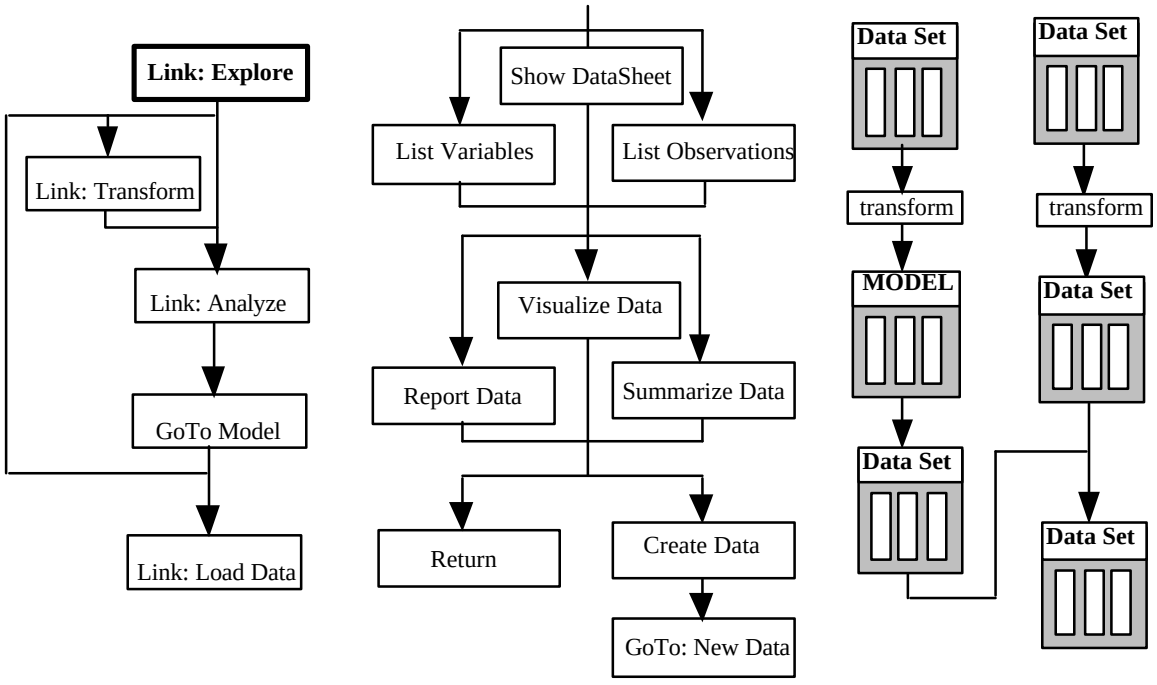


Figure 2-16. GuideMaps and WorkMaps from Young and Lubinsky (1995).

2.2.2.5 Process-Level Interaction Summary

Most of the process-level research uses a graph or tree to represent the process. The process descriptions share many similarities, even though there are differences in the task environments (visualization, database, data analysis). Table 2-2 shows a comparison

of the different approaches to the process-level aspect of database exploration in the visualization, database and data analysis task environments.

author(s)	data structure	purpose	comments
Haber & McNabb	graph (pipeline)	images	simulation data, filtering, little notion of interactions
Upson et al.	graph (cycle)	images and computation	simulation data, filtering, little notion of interactions
Felger et al.	graph	images and interactions	data and configuration / control interactions
Brodlie et al.	tree	images and computation	simulation data, history tree, interaction with process
Canter et al.	graph (network)	database navigation	descriptive graphical indices
Hachem et al.	graph (Petri-Net)	data derivation management.	predetermined derivations
Kersten & deBoer	graph	query optimization	developed browsing session models
Chuieh and Katz	graph	design process management.	history records and design data object relationship determination
Becker & Chambers	audit file and audit plot (graph)	track data analysis process, simple analyses	separate analysis program, object / statement relationships
Carr et al.	tree	manage data analysis process	review and verify process, no analysis data derivations shown in tree
Oldford & Peters	graph (network)	track and analyze data analysis process	causal and analytic relationships, among others
Young & Lubinsky	graph	guide analyses using expert's flowgraphs	application of previous analysis graph to help naive users

Table 2-2. Comparison of approaches to process-level database interactions.

All visualization approaches describe conceptual processes; the approach of Brodlie et al. provides an implementation of a model. Any visualization application must implement the processes of Figure 2-7(a), however. The early models of Haber and

McNabb, and Upson et al., do not discuss interactions, only the somewhat rigid image production process. This is similar to the task taxonomies described earlier. Felger et al. add the notion of interaction with data as well as configuration and control of the process, describing more of the dynamic elements of the visualization process. Brodlie et al. also discuss interaction with the visualization process, by re-using previous simulation parameters in the current context. An interesting note is that the previous visualization work deals with simulation data, not with any database management system. There is no notion of data selection, except for the broad category of filtering.

All of the database approaches use some type of graph data representation, and focus on data management techniques that support interactions and process evolutions. Canter et al.'s conceptual model describes navigations (data location evolution) through a network of similar type nodes, while the others provided implementations. Kersten and deBoer's browsing strategies are also data location evolutions. Other evolutions includes data derivations (Hachem et al.) and design histories (Chuieh and Katz).

The data analysis approaches all focus on managing the entire data analysis process. Data structures range from lists to trees to graphs. Process analysis was performed by Becker and Chambers, Carr et al., and Oldford and Peters. Young and Lubinsky applied a previously constructed process histories to data explorations by a naive user.

2.2.3 Monitoring User-Data Interactions

Monitoring user-data interactions is required to support and exploit the dynamic component of database exploration. Lee (1990) identified seven uses of interaction history: 1) reuse of a previous operation, 2) recording and replaying a script, 3) recovery

from unforeseen errors, 4) navigation, 5) external memory, 6) adaptive interfaces and 7) user modeling. Her research also pointed out the lack of information about interaction patterns and history usage characteristics. Monitoring can either be undertaken by direct observations of users as they interact with data through a data exploration environment, or by automated means incorporated directly into the exploration environment. Whereas direct observations are useful for obtaining the static elements of a process, automatic monitoring is useful for understanding the dynamic elements such as patterns and trends in analyses, and creating adaptive systems.

Direct observation studies are exemplified by the works of Springmeyer, Knapp, Gillan, Wickens and O'Day and Jeffries. Those works primarily serve to identify tasks, though they often do allude to higher-level interaction processes. The automatic retention studies, exemplified by most of the other process-level research, seems to focus on exploration session support and analysis of known tasks.

Exploration session management at the session level is addressed by Nicholson et al., Oldford and Peters, Brodlie et al. and Chiueh and Katz. Exploration session support at the task level is addressed by all of the automatic visualization research, Larson's work on visual browsing and Kersten and de Boer's work on query optimization for database browsing sessions. Database exploration process analysis included Becker and Chambers' investigations of data dependencies, Canter et al.'s navigation characterizations and Oldford and Peters' views of analysis, data flow and causal relationships.

2.3 Data-Centric Aspects of Database Exploration

At the heart of any exploration environment is the data management infrastructure. In order to support database exploration, a software system must minimally possess a

database management component and a visualization component. Their integration is crucial to the effective exploration of large databases. In this section we focus on systems-level issues in database-visualization integration, and survey previous systems-level approaches to supporting database exploration.

2.3.1 The Impedance Mismatch Problem

Integrating relational database and visualization systems requires a unifying data model that "bridges the gap" between the respective system components, and supports the user-data interactions described previously. This gap is often termed an *impedance mismatch* between the data models of the components, and is difficult to overcome due to the intrinsic differences between database and visualization data models.

The relational data model, introduced by Codd (1970), is a formal mathematical model based on a single tabular data structure, the *relation*. Each row, called a *tuple*, is an aggregation of related data values. Each column signifies an attribute, which has a specific data type, or *domain*. Relational databases are collections of relations that have certain constraints to ensure various forms of integrity among the relations. Application data objects are often decomposed into several distinct relations that must be "joined" together to be queried. The *view* mechanism allows a name to be associated with a query, which can be used in other relational expressions. This simplifies querying, lends some object identity to relational processing, and provides a measure of data access security and data independence.

Relational database interaction is almost exclusively through the Structured Query Language, or SQL, which allows the *selection of data based on data associations*. It also supports data update and modification operations, primitive computations on database

attributes, and some reorganization of the tuples making up the query result. SQL provides a limited form of navigation of the query result through a *cursor*, a pointer to the current tuple retrieved from the database. The cursor can be incremented or decremented. Additional tools are usually provided to enhance query result navigation.

Data visualization data models are primarily data structure models, having connectivity and topological specifications, as described in Treinish (1991). Models include lattices (Kao, Bergeron and Sparr, 1995, Hibbard, Dyer and Paul, 1994) and fiber bundles (Haber, Lucas and Collins, 1991). Because visualization is considered by many to be an output process, the only operations described are the filtering, mapping and rendering operations shown in Figure 2-7a. Commercial systems provide analysis operations, and dataflow network visualization construction. They use bulk file loading, and sampling based on topology. Visualization data models are closely associated with research into scientific databases, but they lack data selection operation specifications. Scientific database data models have been implemented for spatial and temporal data, for numerous domain-specific applications.

Thus, relational databases offer powerful data selection over base data stored in relations, but very little data analysis and minimal visualization. They have tuple-oriented interface mechanisms, and are designed for small, frequent data transfers between the database server and client application. In contrast, data visualization systems offer minimal data selection, but powerful display (output) and interaction (input) paradigms for highly structured data. They usually transfer large amounts of data between the visualization data server and the client rendering application, and rarely update or modify data. Not only do the data structural models differ in each system, but the operations supported differ as well. Figure 2-17 shows the impedance mismatch problem. The important questions are

how to construct the database-visualization interface and what operations will be needed to support interactions at the visualization user interface.

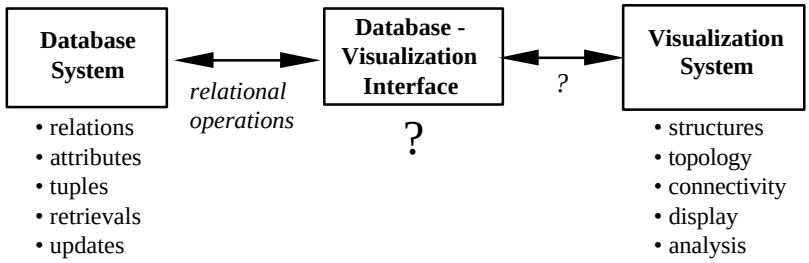


Figure 2-17. The impedance mismatch between database and visualization systems.

2.3.2 The Importance of Metadata

Metadata is a knowledge representation that describes other data, such as a data structure, an analytic process, or a computational environment. Good support for metadata is critical to the integration of database and visualization systems and the support of database exploration operations. Metadata examples include data locations, summary statistics, sensor calibrations, missing data codes, audit trails of processing history and annotations (Treinish, 1991). Metadata can also be visualization rules (Rogowitz and Treinish, 1993), data precision values (Hibbard, Dyer and Paul, 1994), and derivation rules (Hachem et al., 1993). The conceptual model of Kao and Bergeron (1994) includes descriptions for data structures, interrelationships, analysis and validity. A conceptual metadata model for very large spatial databases contains *metadata*, *metaprocess* and *metaenvironment* entities (Trivedi and Smith, 1991). *Metadatabases* for scientific and statistical databases are discussed in Lenz (1994) and Bretherton and Singley (1994). It is clear from the literature that there is no one complete metadata model, and that metadata evolves as it is used.

For database-visualization system integration, metadata is necessary to describe the data structures that need to be transferred between the systems. For database exploration, it is apparent that metadata is necessary to describe the processes that track the user-data interactions.

2.3.3 Integration Strategies

A categorization of proposed database-visualization systems models begun at the 1993 IEEE Workshop on Database Issues for Data Visualization (Lee and Grinstein 1994), and expanded in a subsequent tutorial (Grinstein, Lee and Vasudevan 1994) differentiated several integration strategies. Three strategies were proposed, in increasing degree of integration:

1. *Visualization as a database front end* - The database system is dominant, and the visualization is a simple browser into the query result. Minimal interaction with the query result is provided, mimicing the traditional database forms interface or report generator.

2. *Database as a feature management system* - The database only stores metadata summarizing important features of the data, and the location of the data somewhere within the file system or network. The content-based metadata must first be extracted (requiring KDD techniques), and high-level metadata query tools must be created. Data modeling, as opposed to (1) is much less important.

3. *Database as a visualization backend* - The visualization system is dominant and the database system essentially implements the visualization data model. Three subclasses are identified. In the *transactional file system* model, the visualization system may not know that a DBMS is present, but makes “transactional” data

access statements. In the *model translation* model, a family of strategies that implement multiple data models and storage paradigms is employed, based on how the data will be manipulated. For example, data to be queried at a fine-grained resolution requires a database storage, while more opaque data (e.g., tagged image files) may be stored as in the feature management strategy. In the *abstract rendering machine,* the database data model parallels the visualization and rendering pipelines, supplying a data abstraction for each stage in the process.

Three types of architectures are identified to support these integration strategies, close-coupled, client-server, and distributed asynchronous process communication. In *close-coupled* architectures, all components reside on the same machine (perhaps even in the same process), and the database manages its own visualization. This exhibits higher database performance, but lacks concurrency and distribution. If there are no specialized visualization components or rendering hardware, performance is negatively impacted. In *client-server* architectures, distribution and concurrency are supported, and specialized graphics architectures can be supported, but with increased network communication overhead. An ideal solution is somewhere between these two extremes. In *distrubuted asynchronous process communication architectures*, the database and visualization systems are separated by a manager layer. Database objects are responsible for communicating update requests to their visualization objects.

Another important topic, *data coupling*, was discussed in the context of client-server architectures. Loose data coupling implies query results are maintained and manipulated further in the visualization client without database system intervention. This requires database management facilities to exist at the visualization client. Tight data

coupling implies each interaction involves the database system, and visualization client data management is minimal. Again, the optimal solution is somewhere between these two extremes

2.3.4 Database Exploration Research Systems

Any software system that possesses database, analysis or visualization tools can be considered a data or database exploration environment, and there are hundreds of such systems in the literature and in commercial use. Though our definition of database exploration requires database and visualization components, we relax this criterion on the literature review to show the many dimensions encompassed by databases and visualization. In this section we survey a broad sampling of systems, particularly those that have database and visualization components, or those that specifically target themselves towards visual data exploration (no database component), or those whose focus is to support some aspect of database exploration.

2.3.4.1 Modeling Scientific Experiments

Database exploration often occurs in the context of a scientific experiment, and some research focuses on supporting the entire experiment management process. Cushing, Hansen, Maier and Pu (1993) proposed such a model in computational chemistry by extending an object-oriented database system to accommodate legacy applications, and supporting data and program interoperability in a single unified environment. They developed a “computational proxy” mechanism based on a common computational model and descriptions of an application’s input and output to the database system. Ioannis, Livny, Haber, Miller, Tsatalos and Wiener (1993) developed a desktop experiment life cycle management system to support the design of an experimental study, ordering of

experiments, management of data and analysis of results. They created their own object-oriented database system and query language specifically for scientific experimentation activities. The experiment level of the Gaea system (Hachem et al., 1993, and Figure 2-11) models global change scientific procedures using a graphical dataflow user interface, which is supported by lower levels of derivation semantics and a database system.

Recently, Chen and Markowitz (1995) applied requirements for modeling genomic scientific experiments to the development of new object classes called *protocol* classes, that model experimental laboratory procedures. Their characterization of scientific experiments includes transformations from input resources to output results, sequencing of experiments and composition of experiments. Protocol classes contain input and output attributes, protocol connections, and rules to govern their usage and interrelationships.

2.3.4.2 Database Support for Database Exploration Processes

The interconnected sequences of queries, analyses and visualizations of the database exploration process must be supported by the DBMS. There has been some research into providing database support for sequences of queries, management of query results and optimizations based on available data. Since database searching is a potentially costly operation, data access and management optimizations are required.

Holsheimer and Kersten (1994) propose a two-level architecture consisting of a data mining tool to organize and control the search process, and a parallel database server to give optimal response times for a limited number of query types used by the mining tool. The database is vertically partitioned to target the mining algorithm (classification rule extraction) and the parallel processing DBMS. Kersten and de Boer (1994) introduce a method for optimizing sequences of overlapping queries, stressing the use of partial

results previously extracted. Three browsing strategies were analyzed (see Figure 2-12), and performance measures computed with retention of previous results to be used in subsequent queries. They found their techniques useful for large, broad tables (having many attributes), and for “scientific browsing”, defined as the location of an interesting subset for detailed analysis.

A similar activity to using cached query results in query processing is optimizing queries using *materialized views.* A materialized view is a pre-constructed relation that is stored in the database. One would expect that the database exploration process would create such relations that isolate interesting data for further study. Chaudhuri, Krishnamurthy, Potaminos and Shim (1995) provide a solution for optimizing queries based on materialized views by extending the traditional query optimization algorithm. This cost-based approach can be applied to cached query results, but the authors note that such results are not permanent, in contrast to materialized views. A system table is maintained to organize the queries having cached results and the rules creating the view specifications.

Hamon and Keller (1995) propose a two-level client-side cache of composite object views of relational databases. The lower-level cache contains tuples from each component relation, which are linked together according to joins specified by a Structural Model. The upper-level cache contains the composed objects, using an Object Schema of a single application. This system is meant to support multiple applications, each having a unique Object Schema, to share data in a common relational database.

The FastMap system addresses the processing and visualization of similarity queries, and uses a spatial access method to retrieve data (Faloutsos and Lin 1995). Data objects are mapped into a k-dimensional space such that dissimilarities are preserved.

Feature extraction functions (devised by a domain expert) were used for each dimension, and displays consist of simple 2D and 3D point clouds.

2.3.4.3 Interaction Support for Database Exploration Processes

The few interaction support issues mentioned in the literature revolve around allowing user input to be specified to the application objects.

Mamou and Medeiros (1991) create a view support mechanism for interacting with object-oriented databases that combine visual presentation with the data specification, called a *hyper-view*. The visual presentation provides complementary information related to graphical display and interaction of the underlying data view. Interactions include navigations based on view access privileges, visualizations of the data view, data selections from the view. Function specifications include masking (column projection), representation mapping, and various display parameter settings. This system is used for generating user interfaces for an object-oriented database system.

Felger and Schröeder (1992) proposed the *visualization input pipeline* (VIP) to facilitate “semantic interaction”, or interaction with application data (see Figure 2-8), not the visualization process. They develop an input-response cycle that defines a dialog layer between the traditional visualization output pipeline (VOP) of Figure 2-7(a), the VIP and the user / display, and mapped the display cursor to the application data. The VIP consists of inverse mapping modules of the VOP. Their inversion methods are defined as: 1) an inversion function if the VOP module was a bijection, 2) a lookup procedure, and 3) a listening procedure that forces a re-execution of the VOP module, to generate the required data. Four dataflow architectures are discussed.

Zhou and Kubita (1992) developed an object-oriented user interface model for a computer-aided design (CAD) system. This model adds interaction resolution to the definition of the user interface, in addition to the traditional interface visual component specifications. This model takes a layered approach to the interface, and included *activators* to supply user input, *mediators* to resolve activator input based on application semantics (such as object collisions) and an *application exposure* member function interface. Mediators sequence the user interface actions and forward messages to application objects, incorporating a form of intelligence based on interactions and object semantics.

The *dynamic query* paradigm is a method for rapidly manipulating the visualization of a data set, using range sliders (Ahlberg, Williamson and Schneiderman 1992). This paradigm allows interactive range queries on a multidimensional data set. It is useful for quickly obtaining an overview of the data, and probing along the dimensions of the attributes. Ahlberg and Wistrand (1995) describe a system that constructs a dynamic query application automatically from database tables. This system also allows visualization operations such as panning, zooming, spatial rotations for 3D scatterplots, filtering and selection of detail data on specific data set elements.

Many interaction and visualization tools have come out of the Xerox Palo Alto Research Center (PARC), and are worth mentioning. Most of them relate to displaying data, while making better use of valuable display space, such as the Perspective Wall (Mackinlay, Robertson and Card 1991) and Cone Trees (Robertson, Mackinlay and Card 1991). Others focus on interacting with data through direct manipulation interfaces, such as Magic Lenses (Stone, Fishkin and Bier 1994) and Table Lenses (Rao and Card 1994, Pirolli and Rao 1996).

2.3.4.4 Knowledge Discovery Systems

Knowledge discovery systems stress the automatic discovery of knowledge, such as rules and classifications, from data sets or databases. Matheus, Chan and Piatetsky-Shapiro (1993) gave an overview of knowledge discovery systems. They define the following components as part of a prototypical knowledge discovery system: a *controller* to direct the invocation and parameterization of the other components, a *database interface* to generate and process queries, a *knowledge base* to hold domain information, a *focus* component to determine the portion of the database to analyze, a *pattern extractor* and an *evaluator* to determine the utility of extracted patterns. This model was used to evaluate systems based on versatility (the range of application domains and types of discoveries achievable) and autonomy (the degree of freedom from human direction). They observed a characteristic tradeoff between the two dimensions, shown in Figure 2-18. More autonomous systems rely heavily on domain knowledge to perform very specific tasks, while more versatile systems rely heavily on the user to direct a large number of discovery tasks. The "ideal" KDD system is both highly versatile and autonomous, a feat that has yet to be realized in practice.

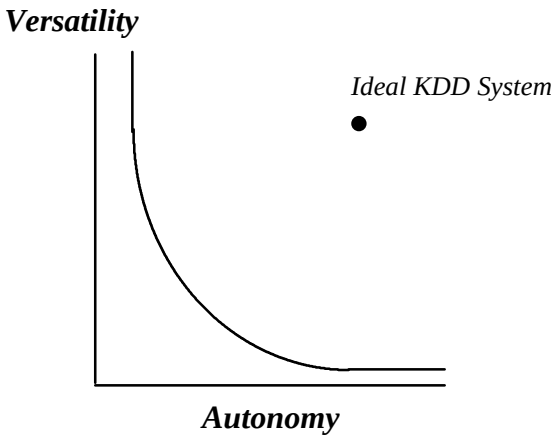


Figure 2-18. KDD system autonomy versus versatility, from Matheus et al. (1993).

The IMACS system (Brachman et al., 1993) was developed to support the iterative nature of the discovery process, framed around the formulation and validation of hypotheses about a data set. It supports a domain knowledge representation model that is extended by the user as an exploration progresses. This knowledge base stores information about domain objects, their relationships and high level concepts that describe a data set. Entire database tables are pre-loaded, reuse of queries and their results is allowed, and discovery tasks (segmenting data via queries, forms and graphs, and viewing data as tabular descriptions and associated graphs) are performed completely within IMACS. They describe segmented data as “first class interface objects” that can be analyzed with all viewing and segmenting operations.

The Recon system (Simonudis, Livezey and Kerber 1996) utilizes rule induction, deductive databases and data visualization cooperatively to create rule-based models from relational databases. Multiple visualization (scatterplots and histograms) are initially used to obtain an overview of the data and highlight promising areas for study. The visualization capabilities include marking subsets of interest, focusing on a portion of the display (zooming), and selecting a subset for an alternative visualization or export to an analysis module. The deductive database allows the analyst to express knowledge constructs and test the database against them. The rule induction system automatically explores the database to determine any embedded rules by continual modification of existing rules.

The SKICAT system (Fayyad, Djorgorski and Weir 1996) applies supervised classification learning techniques to automatically reduce and analyze a large astronomical data set. This work required the integration of image processing, data classification and database management. The prime motivation for using automated techniques was the size

of the data set - over 3 terabytes of images containing over 2 billion celestial objects that had to be reduced to catalog entries based on 40 attributes determined from the image data.

2.3.4.5 Integrated Database-Visualization Systems

There have been several database-visualization integrations in the literature, largely as a result of the NSF scientific database initiative. Most of those projects deal with managing spatio-temporal data, with little emphasis on integrating visualization or interaction aspects. Examples of those that do focus on the visualization component include medical database systems using imagery data, and earth science databases using 3D spatial data. We review those systems, and additional systems that focus on the visualization integration problem.

An early system that provides tools for managing discipline-independent simulation and visualization data uses the Common Data Format (CDF) in an indexed flat file, plus one binary, random-access file per variable (Farrell, Appino, Foty and Linton 1991). *Data sets* within the database are retrieved and sorted by a number of attributes, including name and keyword. Only one attribute is used per query, and no predicate capabilities are offered.

The KMeD system integrates two distinct data models into an object-oriented database for medical research (Cardenas, Taira, Chu and Breant 1993). One data model is based on stacks of medical images, and the other on the temporal-evolutionary changes that can be deduced among images. Processes such as bone growth and disease spread can be queried, in addition to the traditional patient information and spatial relationships between objects.

The QBISM system is a prototype tool that allowed visualization and querying of 3D human brain scans (Arya, Cody, Faloutsos, Richardson and Toga 1994). It adds a 3D scalar field data type to an extendible DBMS, and develops spatial operators to manipulate these entities efficiently at the DBMS server. The extended database was then integrated with a dataflow visualization environment, where the analyst could specify queries as dataflow graphs, visualize results, and perform visual operations (rotations and slicing planes) on the query results.

The FLEXIDESC system combines a flow-based visualization front-end with interfaces to statistical analysis tools and a database interface to a scientific data model for earth science research (Sparr, Bergeron and Meeker 1993). Queries are expressed as dataflow networks that created new derived data. The underlying scientific data model is based on lattices (Kao, Bergeron and Sparr 1993, 1995), and includes provisions for data structure, *syntactic* metadata for descriptions and *semantic* metadata for relationships. The focus is to develop the new scientific data model, and provide an environment for researchers from different disciplines to collaborate on projects.

The Sequoia 2000 project is a large-scale visualization-database project for global change research (Stonebraker and Frew 1993). Among its objectives were large, fast storage, a central DBMS for data sharing and maintenance, integrated visualization tools able to handle data from divergent sources and high-speed networking such that data can be accessed off-site. Several visualization environments have been developed, which we now describe.

An early Sequoia prototype combines the Postgres extended relational database system with the AVS visualization environment (Kochevar, Ahmed, Shade and Sharp 1993). This simple visualization management system allows data of interest to be located

from a spatial range query over an image. All visualization information is stored within each data set. Database data is visualized automatically by downloading the appropriate visualization script or program.

The Tioga system creates a dataflow visualization environment using Postgres as the database backend (Stonebraker, Chen, Nathan, Paxson, Su and Wu 1993). This system combines user-defined data types and functions, and develops multidimensional access functions. A flight simulator paradigm is used in *browsers*, which are DBMS application programs, to navigate query results. *Abstracts* of varying resolutions of data are provided to support zooming operations and performance improvements. This evolved into the Tioga2 system, which implements direct manipulation techniques to build applications visually (Aiken, Chen, Stonebraker and Woodruff 1996). A small set of primitive operations for transforming data and its visualization are provided to ease the complexity of developing applications.

Visualizing relational databases has received some attention in recent years, and our concern is with techniques that surpass the simple graphs and plots provided by commercial products. Relational databases entail a special problem in that the data is often not spatial, so novel visualization techniques have to be developed.

The DataSpace system uses a 3D graph layout paradigm with graphical operators to interactively visualize and navigate through large databases (Anupam, Dar, Leibfried and Petajan 1995). Data extracted using a form is mapped to nodes and edges of the graph. Nodes contain a 2D array of information panels that each hold a 2D graph of an attribute value over time. Nodes are then assigned a position value for 3D display, based on reducing the number of edge crossings and edge lengths. The 3D graph can then be

manipulated by the analyst. Zooming, navigation among graph nodes and transparency operations are supported.

The VisDB system provides query relevance feedback for large databases (Keim, Kriegel and Seidl, 1993). Each result tuple attribute is ranked, according to its relevance to the query, and each tuple is mapped to a single pixel based on these rankings. The analyst can then see several displays of the query based on the conditionals specified, and how well the query satisfied each conditional. Numerous display schemes have been implemented (Keim, Kriegel and Ankerst 1995). By mapping a tuple to a single pixel, up to one million tuples can be displayed at once.

2.3.4.6 Visual Data Exploration Systems

Visual data exploration systems do not have a database component. They rely on the presentation of information visually to guide the exploration of a data set. These research systems stress the interactive manipulation of data representations and visual tools to configure and probe the data visualization for values and structure. They are primarily file-based, and often load entire data sets into memory for visual manipulations.

The Exvis system uses iconographic representations of data to create texture displays of large data sets (Pickett and Grinstein 1988, Smith, Grinstein and Pickett 1991, Grinstein, Sieg, Smith and Williams 1992). Icons consisted of “families” of lines and color fields that can have multiple data parameters mapped to them. 2D and 3D icons have been implemented, as well as many interactive tools to configure the display, the data mappings, and multidimensional probing of the data. *Brushing* (highlighting visualization components with the cursor - see Becker and Chambers (1987) for an overview) can be mapped onto

sound to further increase the output data dimensionality. Up to 15 data parameters have been mapped to display attributes simultaneously.

The XGobi system performs multidimensional visualizations by combining algorithmic projection methods, such as projection pursuit and data-space rotations, with simple visual displays, such as scatterplots (Buja, Swayne and Cook 1996). The analyst can link multiple displays together, and see the effect that brushing one display has upon the linked display (Buja, McDonald, Michalak and Stuetzle 1991).

The Xmdv system integrates several common multidimensional visualization methods, including scatterplots, glyphs, parallel coordinates and hierarchical techniques (Ward 1994). Users can see their data visualized in different ways simultaneously and perform multidimensional brushing across the displays (Martin and Ward 1995). A similar system, called VisuLab (Schmid and Hinterberger 1994), combines scatterplot matrices, parallel coordinates, permutation matrices and Andrews curves for comparative visualization.

The IDES system contains a framework that supports three kinds of data exploration subtasks: visualization operations, data manipulation operations and data analysis operations (Goldstein, Roth, Kolojejchick and Matttis 1994). It uses automatic visualization techniques to assist in rendering tasks, and two interaction mechanisms. It employs dynamic queries and aggregate manipulators (Goldstein and Roth 1994) to control the level of detail of data visualized.

2.3.4.7 Summary of Research Systems

The research systems all focus (understandably) on data processing as opposed to direct user support. Systems issues are addressed at the task level, however, so the user is implicit in just about all of the prior research.

The scientific database research focuses primarily on modeling scientific data such as spatial and temporal data. The scientific process management efforts model the task environment. In those research efforts, data models are developed using relational, extended-relational and object-oriented database technology. Database support focuses on efficiency improvements for querying (browsing, materialized views, system architectures, etc.), and more expressive querying over these new data types. Other important issues include object views of relational data, and the use of object-oriented technologies for expressiveness.

Interaction support looks primarily at user-visualization support capabilities, such as specifying the visualization with the data view (hyper-views), dataflow data probes (the visualization input pipeline), object-oriented user interface layering to incorporate application knowledge to resolve user input, and data probing with dynamic queries.

The Knowledge Discovery in Databases research notes important tradeoffs in system capabilities - the versatility versus autonomy graph of Figure 2-18. The two exploratory systems described support numerous tasks (query, analysis and visualization), but focus on the analytic tools; the visualization is used to guide the exploration and to show clusters. In contrast, the automated system described focuses on a narrow, well-defined task.

The integrated database-visualization systems are relatively recent developments. They deal largely with the impedance mismatch issues, scientific data modeling, and how to visualize queries and their results. In contrast, their predecessors, the visual data exploration systems, are more concerned with multidimensional visualization and interaction issues such as brushing and visual data probes.

With respect to the integration strategies described in Section 2.3.3, most approaches have occurred in the database realm, and subscribe to the *visualization as a database front end* model. Most systems do offer some visual interaction capabilities with the database, however, though queries are often textual. The Recon system and dynamic query systems offer simple visual interactions with graphical ranges-of-interest and slider widgets, respectively. The KMeD system provides new data models and an advanced query language, but offers little visual interaction with query results that are images.

In contrast, the QBISM, Tioga and FLEXIDESC systems seem to follow the *database as a visualization backend* model, as new data models and visual query paradigms are employed. These systems all model queries as visual dataflow networks. The *abstract* concept proposed for use in the Tioga system is similar to the *database as a feature management system* model.

2.4 Literature Review Summary

In this chapter, we have presented an extensive review of the literature related to the multitude of aspects of database exploration. We have surveyed the varied definitions of what we call database exploration, and noticed many similarities among definitions. Similar themes emerge across the different domains defining database exploration, the

slight differences being attributable to the goals of each domain (such as statistics-oriented terminology in the case of Velleman).

We have surveyed user-centric aspects of the database exploration process, specifically the elemental tasks that form the vocabulary of database exploration, and the process models that connect the tasks. Several task enumerations have been proposed, with a good deal of overlap across domains, again differing due to the domain goals. The most useful process models are based on trees, networks or graphs, as opposed to linear lists. Of the proposed models, it seems that the statisticians have developed the most striking and descriptive examples.

Finally, we have surveyed data-centric aspects of the database exploration process, specifically the impedance mismatch between database and analysis systems, the importance of metadata to address impedance mismatches, database-visualization integration strategies and numerous research systems in database, knowledge discovery, statistics and visualization domains.

From this survey, it is evident that the universe of database exploration spans many domains (as it should be), and that there are many different (and legitimate) approaches to database exploration, depending on the goals of the domain.