

# Contextual Data Processing

Victor E. Hansen, MKS Inc.

“It ought to be remembered that there is nothing more difficult to take in hand, more perilous to conduct, or more uncertain in its success, than to take the lead in the introduction of a new order of things. Because the innovator has for enemies all those who have done well under the old conditions, and lukewarm defenders in those who may do well under the new.”

Niccolo Machiavelli  
*The Prince*, 1513

## Abstract

Advances in information technology are flooding us with data. Current data models are no longer adequate for obtaining the information we need from this data. We propose that making information systems more human-like will make interacting with humans more effective. Numerous ways in which humans and information systems differ are discussed. V4, a contextual multidimensional data model, is introduced followed by examples of V4's use in several problem domains.

## Introduction

Technology is allowing us to store ever greater amounts of data. Not only can we store more data but more types of data. Business data, sales data from grocery scanners, music in the form of CDs, pictures, sounds, books, even television is going digital with the acceptance of the new standards and HDTV. All of this is becoming more and more available through the World Wide Web and the Internet. In some instances we can actually get at the data and answers we seek, but as the quantities and types of data increase, getting answers becomes difficult to impossible.

Solutions to the data problem will come about only through novel ideas and perspectives. Our premise is that to make data more accessible we must first enable it with *human-like* qualities. This calls for new paradigms in both the organization and manipulation of data.

To this end we have developed a theoretical model for describing and accessing data. The model incorporates several new ideas, which vastly simplify the problems associated with describing and analyzing large databases. The two basic principles of this model are *context* and *multidimensionality*- two qualities inherent in the way people think. We have developed a product, V4, based on these concepts and have been able to use it to elegantly solve many difficult problems.

This white paper describes the ideas and characteristics of V4. First, however, we look at ways that people view data (or knowledge) and how these views differ from the operating characteristics of database systems. Next we examine some of the characteristics of V4 to show how it can enable users to better access data by modeling and manipulating the data in a form that is much closer to how people think about data. Problem domains well suited for V4 are reviewed. The paper closes with an update of V4's implementation status and current applications.

## How People Differ from Database Systems

We make no claim as to knowing how people actually think. We can be fairly safe in saying that people can and do represent, store, and think about data in ways that no commercial database product can begin to do. Some of the major ways in which people think and react differently from database systems are discussed in the following section.

### **Contextual**

We believe that the biggest difference between people's memory and databases is the notion of context. People interpret *everything* within a context. Words, phrases, actions, intentions are all “evaluated” (and colored) by the *who*, *what*, *where*, and *when*. Not only do external events and situations affect context but people's internal ideas and moods also exert a great influence on how everything is interpreted.

Context allows us to easily do things that are very difficult to do with computer systems. People apply the same general rules in different contexts. People pretend and play make-believe. People communicate

## Contextual Data Processing

the same information in different forms for different contexts (e.g. explaining a computer to a child versus an adult).

### ***What versus How***

Rarely when a person wants an answer to a question do they want to know how the answer was obtained (“Tell me the time, not how to make a watch”). This is usually not the case with requests to an information system. Details about the structure of the databases, how they interrelate, and specific field names are all necessary to obtain answers.

### ***Generalize with Exceptions***

A typical pattern of learning is to first use a general rule (“*i* before *e*”) and then learn the exceptions (“except after *c*”). People are very good at this; to the point that they can recite the general rules but not be able to tell you the exceptions until an exception happens. Databases are not good at dealing with this type of situation. Databases store data, not general rules.

### ***Facts, Rules, Procedures***

If you are talking to someone on the phone and he asks you for your home phone number you can recite it right off. If he asks for the number of the local pizza parlor, which you don’t know, but have right by the phone, you can recite it back to him. The person on the phone would think that you knew both numbers. The reality is that you only *knew* the first but could *calculate* the number for the pizza quickly enough so that the person on the phone could not tell.

Knowing how to get the answer is almost as good as knowing the answer. The way we store information in our heads makes the boundary between *knowing* and *knowing how* seamless.

### ***Facts About Themselves (Metadata)***

People know facts and lots of *stuff* about the facts. Most people can find a phone number in a phone book. They can also tell you about the phone book, how it is organized, and what the different codes and abbreviations mean. This type of knowledge is known as metaknowledge, or when applied to data in information systems, metadata. Newer information systems incorporate metadata but usually as a separate component, i.e. data and metadata are rarely intermixed.

### ***Multidimensional***

We have the ability to store knowledge by varying parameters or dimensions. Ask a person if a TV show is any good and you might get a response like, “The last show was pretty bad, but in general it’s OK. My wife likes it, but the kids don’t.” In this response we have information about the show’s quality by time, by specific show versus most shows, and by individuals. How would you represent this in a database?

The real world is varyingly multidimensional. By that we mean that not only is it multidimensional but that the number of dimensions associated with a data element varies. How many factors or dimensions go into the price of an airline ticket these days, and which factors apply when you purchase a ticket for business and which apply when you purchase a ticket for a family vacation?

Relational databases are essentially three dimensional- tables, rows and columns. Additional dimensions are created by duplicating tables, rows, or columns. The techniques and commands to access data vary depending on how the data is embedded in a particular row, column, and table. Determining how to represent all the dimensions associated with data is one of the major challenges for a database administrator/designer. The task of adding dimensions once a system has been up and running can be next to impossible. For example, if a pricing database has been designed with customer, date, item, and quantity as the components (or dimensions) of the price of items and now the sales rep commission must be added as an additional component. What databases have to change, what procedures, what reports?

### ***People Forget***

“Use it or lose it” as the saying goes. People forget facts that are not used. How much of your grade school, high school, and college education do you really remember? Databases do not forget yet studies have shown that data quality suffers greatly over time if the data is not used. This raises interesting issues for the design and long term use of data warehousing.

## Contextual Data Processing

### Summary

Why bother with these differences? Because incorporating some of our human abilities into information systems would result in many benefits. A very brief summary is given below.

- **Contextual** – A contextual system would be able to interpret requests and data according to the requestor (for example *year-to-date* might mean seasonal to a marketing manager, and fiscal to a financial person). Answers would be given in a format appropriate to the requestor (different languages, different levels of detail according to the expertise of the requestor). Context would also allow users to set up multiple what-if or pretend scenarios to examine and manipulate data and models in different situations (What if an x% decrease in price resulted in a y% increase in sales? How would that affect the bottom line, salesrep commissions, customer volume discounts?)
- **What versus How** – Being able to request what data you want without having to specify how to get it removes the underlying database structure from the users. People should not have to know the gruesome details of a data warehouse in order to access the data within it.
- **Generalize with Exceptions** – The ability to store general rules and exceptions would make many databases much smaller, simpler, and easier to understand. For instance, a pricing database may have thousands of records (one for each inventory item), when in fact, the prices could have been represented with something as simple as “all prices are 20% over cost except item.xyz which is \$4.98 and abc which is \$10.40.”
- **Facts, Rules, Procedures** – An information system must be able to manipulate facts, rules, and procedures in a seamless fashion in order to provide the high level of integration needed to answer real-world inquiries. It is no longer sufficient to simply report back data or sums of data; financial, statistical, and industry specific models and massaging must be performed.
- **Metadata** – An information system that “knew” all about itself, when coupled with the ability to incorporate procedures, should be able to automatically generate low level database queries in response to a high level user request (i.e. allow the user to ask what, not specify how).
- **Multidimensional** – An information system designed around a multidimensional paradigm would allow end-users much greater freedom to access and interrelate data. Users would not be tied to the relational table-row-column structure of the underlying data.
- **Forgetting** – The point here is not to have information systems be able to “forget” but to ensure that the data residing in the system is accessed in a timely manner to ensure its quality.

### The V4 Model

The V4 model is a new paradigm for the representation of information systems. V4 views the world through a contextual multidimensional model. It is *not* based on the concepts of tables, rows, and columns found in relational systems. It is *not* based on artificial distinctions between data, metadata, and procedures. V4 is a complete information system model that enables you to integrate data from multiple sources, manipulate this data, and report on it- all in a contextual framework. The remainder of this section gives a brief overview of the important V4 concepts.

#### **Multidimensional**

V4 is based on the concepts of dimensions and points contained within a dimension. In V4, a **dimension** represents a concept and **points** along that dimensional line represent instances of that concept. For example an Integer dimension would have points representing the integer numbers. A color dimension would have points representing the different colors. Dimensions can be used to represent any concept so we can have a customer dimension where each point represents a customer. Or a flight dimension where each point represents a scheduled airline flight. At a more abstract level, we may also have a string dimension where each point represents a text string or a C-Program dimension where each point represents a C language program.

Points, unto themselves, do not carry any information. Information is represented as the combination or **intersection** of points. For instance, the point representing inventory item xyz, by itself, has no information content. Nor does the point representing the concept of “price of”. But if we put them together, or intersect them in the multidimensional space, then we get another point representing “the price of item xyz” which we can assign or **bind** a dollar value which is itself just another point. In V4 we bind value points to the intersection of other points. Facts are represented by binding values to the intersections of specific points as

## Contextual Data Processing

in “the price of item *xyz* is \$9.99”. Rules are represented by binding values to the intersections of point ranges like “the price of all items is 20% over cost”. Remember that procedures are points so the formula “20% over cost” is another point. V4 allows us to have specific facts and rules, which may overlap as in the two examples just cited. If we were to ask for the price of an item, V4 would see that there were two possible intersections and take the *best* one. In this example the best one would result in \$9.99 for item *xyz* and “20% over cost” for all others.

V4 uses sophisticated rules to determine which of several possible intersections is the *best* match. In this pricing example intersections such as “the price of item *xyz*”, “the price of all items”, “the price of item *xyz* on 10-Jan-98” and “the price of all items sold to customer *abc* during 1998” can all coexist and result in different answers. The users and designers of V4 applications do not have to be concerned with mechanisms underlying this representation – it is all performed by the V4 runtime system.

### Contextual

V4’s contextual nature works hand-in-hand with the multidimensional model. The V4 context is simply a collection of points maintained as an independent set. V4 uses the context to determine the best answer when presented with a query. Again, using the above example, if the query were stated as “the price of item *xyz*?”, V4 would find the best intersection of the point for “price of” with the point for item *xyz* plus any points in the context. If the context contained the point “10-Jan-98”, even though the query only specified “the price of item *xyz*”, the result would be the intersection of “the price of item *xyz* on 10-Jan-98”. This is because, in the current context, it is the *best* match. Similarly, if the context also included the point for customer *abc* then the best intersection for the query would be “the price of all items sold to customer *abc* during 1998”.

The power of this approach lies in the ability to design implicit V4 procedures, which become explicit when executed or evaluated in a particular context. This means that generic rules can be specified once and then used in many different contexts.

Context is used to maintain two types of points. The first, as demonstrated in the above examples, is to use context to reflect the current state. Context points of this type include the date, the user, the type of request, or anything else reflecting the current environment and query.

The second class of points is for declaring a *pretend* or *what-if* scenario. By adding contextual scenario points we can say “What if ...?” or “Let’s assume that ...”. Expanding on the pricing examples above, we could make the binding “In scenario *A*, the price of item *xyz* for customer *bcd* is \$7.99”. Evaluating the query “price of item *xyz*” in the context of customer *bcd* would normally result in \$9.99. However if “Scenario *A*” were also in the context then the “price of item *xyz*” would result in “\$7.99”. Context can be used for more interesting examples- “In scenario *B*, the price of *xyz* is 10% less than normal”. Here, “normal” refers to whatever the result would have been without scenario *B*.

### V4 Primitives

Many standard routines found in statistical, financial, and general programming environments are provided by the V4 environment in the form of predefined primitives.

- Time – Time can be modeled in many different ways with many different dimensions. Primitives are provided to quickly and accurately convert among the many different modes of time.
- Statistical – All of the standard distributions, inverses, functions, and tests are available directly within V4. These functions combined with V4’s ability to manipulate time make it an extremely powerful statistical analyzer.
- Financial – Many high level financial routines are provided including those for the time-value-of-money, coupons, securities, and treasury bills.
- List Processing – Lists are an important construct within V4. Primitives are provided to enumerate through lists (e.g. all points on a dimension), to sort lists, to select subsets of lists, and to merge lists. “Lazy lists” can be defined to efficiently enumerate through extremely large lists.
- Database – Large data sets can be handled in several ways within V4. For optimum speed, the data can be loaded directly into V4. Where there is simply too much information or the up-to-date accuracy of the data is critical, V4 can access data through ODBC interface primitives. Non-ODBC databases can be access through a text table handler.

## Contextual Data Processing

- Dimensional – Much of the data to be analyzed by V4 is not in a multidimensional format. V4 primitives can be used to quickly scan flat data (upwards of 20,000 rows per second) to translate it temporarily or permanently into a multidimensional form.
- V4 – Primitives within V4 can be used to construct new V4 dimensions, points, intersections, and bindings. This enables V4 to construct new rules, as it needs them. For example, metadata information can be used to convert a high-level user request into the specific rules required to access, aggregate, analyze, and report back the answers.

## Problem Domains for V4

Very little research has been done in contextual systems although recent interest seems to be growing. Context can be used in many different problem areas. Robotics, reasoning systems, expert systems, natural language processing, are all prime candidates for contextual models. Below we will limit the discussion to the use of context in business database management. Note that all of the examples given below were generated from within Microsoft Excel using a V4 add-in.

### Financials

V4 has many characteristics making it ideal for financial analysis and reporting. Dimensions can be defined for company, department, profit center, general ledger account, forecast, budget, currency, etc. Reports for any combination of dimensions can be easily generated. New budgets can be quickly formulated with general rules and filled in by department heads with specific facts. Different revisions can be maintained. Models (see below) can be developed and then run on any subset of the data. Econometric data from other sources can be compared and V4's financial primitives can be used to compare and optimize.

Monthly Actuals for Year 1996 Department 2292				
Account	Description	January	February	March
6100	SALARIES- OFFICE	20,941	23,768	25,241
		38,300	38,300	38,300
6120	EMPLOYEE BENEFITS EXPENSE	5,444	6,586	6,941
		10,500	10,500	10,500
6330	PROMOTIONAL EXPENSE	426	0	0
		0	0	0
6335	TRAVEL EXPENSE	1,096	9,382	5,092
		2,000	2,000	2,000

The above report shows actual versus budget for department "2292". The first line for each expense account is the actual for the period. The line below is the corresponding budget amount. If the actual expense exceeds the budget the actual amount is highlighted in red. The report (with formatting) was generated with less than ten V4 rules.

## Contextual Data Processing

### Sales Analysis

Sales data can be captured in flat files (as is typical in relational transaction processors) and then viewed multidimensionally within V4. Sales by any time component, geographic area, sales territory, inventory attribute, vendor, or customer data can be easily and quickly generated. Statistical routines can be run to project future demand. Results of any analysis are immediately available for additional financial reporting or manufacturing modeling. Sales figures can be altered within what-if scenarios giving executives the opportunity to view possible management decision outcomes.

	1996Q1	1996Q1-\$	1996Q2	1996Q2-\$	1996Q3	1996Q3-\$	1996Q4	1996Q4-\$
ANGEL	JOHCH	62,826	JOHCH	65,190	<i>STRON</i>	65,966	JOHCH	65,763
	HIGHA	24,470	MELIJR	47,393	MENAR	20,610	GRAYR	51,499
	POLLO	17,482	WEIDA	17,103	POLLO	12,893	MELIJR	45,819
BARRJ	SHAPI	48,018	REFFI	32,722	REFFI	6,289	REFFI	12,676
	REFFI	29,557	SHAPI	13,369	SHAPI	5,471	SHAPI	3,382
BROWL	<i>RUCKS</i>	98,130	MAHEM	52,392	METZL	31,577	METZL	33,945
	MAHEM	39,779	TUTTGL	33,000	GRIEM	23,807	RALPH	30,858
	HUCKE	35,306	FUHR	30,930	TUTTGL	19,516	WEEKS	30,387

The above report is an excerpt of a “top sales rep by manager” report. It lists for each sales manager, the top three sales reps and sales for each of the four quarters in 1996. The top rep for each quarter (across all managers) is highlighted in red italics. This report was generated from “flat” sales order detail data with less than ten V4 rules (five rules for the data, four for formatting).

V4 typically has low level transaction data available. This can often be used for dynamic drill-down. High level results can be quickly displayed followed with more specific drill-downs for supporting detail.

### Data Quality Models and Assurance

As it becomes easier and cheaper to collect more and more data, the tendency to “grab it now because we may need it later” becomes more prevalent. The less this data is used, the more dubious its value and quality. V4 can be utilized to summarize, correlate, tabulate, and point out trends in large amounts of data helping to ensure its quality. Metadata can be used to ensure that interrelationships are meaningful and that all collected data in a warehouse is being reviewed in some fashion.

### Metadata Repository

Metadata is literally “data about data”. It is more than just the mechanics of a database; it describes what we *know* about the components of the database. What one person *knows* about something might not be exactly the same as what another person *knows*. Necessary requirements for a state-of-the-art metadata repository include being able to represent- different points of view about a database, how the database has changed over time, and how different systems reference the data. A contextual multidimensional space can incorporate these different points of view and reference them accordingly.

Given a metadata repository, the next step is using that information in assisting users. For instance, using the information to navigate a complex data warehouse. Again, a multidimensional model with the ability to incorporate procedures easily allows this.

### Security Models

Some IS managers view database security as a subset of a metadata repository, others think it deserves its own classification. In either case, security is about allowing or denying access to data *in various contexts*. The V4 model allows the security administrator to initially set up general access rules and then add more specific access rules as situations warrant. The general rules can either permit access (with subsequent rules denying as needed) or deny access (with subsequent rules allowing as needed). In either case new contexts can be added with time without having to upset any existing security policies.

## Contextual Data Processing

### Manufacturing Models

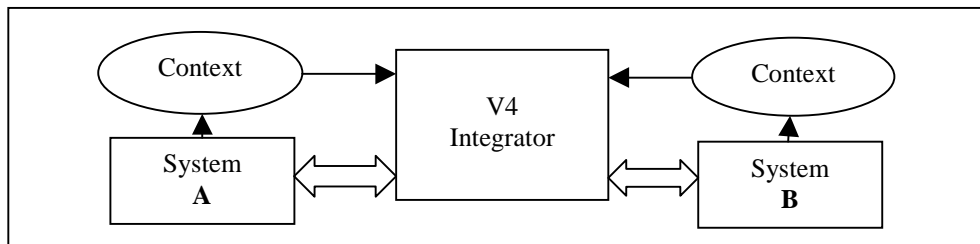
Complex models can be easily developed, maintained, and evaluated with V4. Rules can be changed with a simple change of context. Data from a variety of sources can be run through the model again by a simple change of context. The example below shows the Excel output of a production-scheduling model. The input data for this spreadsheet was taken from a production schedule matrix. Raw materials were exploded through a bill of materials file. Finally actual onhand and expected purchase order receipts were merged. The entire model, explosion, aggregation, and exporting to Excel was done with less than 15 V4 rules (eight to perform all of the data manipulation from flat files, six to format for Excel).

Id	Description	Receipts	OnHand	Needed 2/2/98	Needed 2/9/98	Short 2/2/98	Short 2/9/98
SKU10067I	CANDY, SUGAR FREE HARD BULK	0	4,666	0	0		
SKU10250D	SHRINK BANDS PO# 72679 on 05-Feb-98 for 15000 PO# 72730 on 15-Dec-98 for 5000	20,000	0	0	37,080		37,080
SKU10258G	LBL-HADD PNUT CRUNCH 6 OZ	0	0	12,360	37,080	12,360	37,080
SKU10264I	FLAVOR BUTTER PLUS EMULSION	0	0	0	0		
SKU10266I	CANDY GOLD FOIL CROQUETTES	0	1,333	50	60		
SKU10278I	NUTS, FANCY MED PECAN PCS	0	994	0	1,896		902
SKU10316C	15 CAV OLD FASH CRML	25,000	6,729	0	0		

The color-coding indicates problem areas. Red items indicate that onhand quantities do not cover production needs. Green items indicate that onhand quantities are insufficient but that expected receipts should cover the shortfall.

### Inter-Systems Data Flow

V4's contextual nature makes it an excellent choice for integrating and coordinating the data flow among different (legacy) systems. V4's powerful data manipulation capabilities and internal primitives for translating data types simplify the task of implementation.



The above flow chart represents two aviation systems. Each references an *altitude*. For system A altitude may be defined as feet above sea level, for system B it may be defined as meters above ground level. V4 can reside between these two systems and translate altitude to the value appropriate to each system's context.

## Contextual Data Processing

### Conclusions

#### ***What V4 Is and Is Not***

V4 is not a canned package. It does not provide its own graphical user interface. It is not a tool to be given *directly* to end-users. V4 is a powerful new tool for maintaining, analyzing, manipulating, and reporting data. V4 is to be used behind the scenes in one of the following ways-

- As one of a set of tools for a turnkey solution. V4 integrates well with relational data and can be integrated with many different front and back-end products.
- As a background data processor for Excel. An Excel add-in has been developed giving Excel users the ability to directly access V4. Macros can be used to guide users through any necessary setups. Specific routines can be easily incorporated into the Excel menu structure. Users can perform what-if analysis and drill-down operations directly from within Excel.
- As a semi-natural language processor of user requests. Data, metadata, and procedures can all be incorporated into V4. High level user requests can be converted directly into low level V4 rules (commands) with relatively simple generic V4 rules. Flexible ad-hoc access to complex data warehouses can be (and has been) easily accomplished.

#### ***Implementation Status***

V4 has been under development for approximately seven years. Patent applications were applied for key aspects of the technology in 1993 and have been recently approved. Currently, V4 runs on most major Unix platforms, VMS, and WindowsNT. An add-on dynamic link library has been developed for Excel and is used for interactive access. V4 has been used in a variety of applications ranging from profit center analysis, to budgeting, to sales forecasting. Current models are in the hundreds of millions data point range.

Performance is measured two ways. The first is the size of the data models. The models are typically loaded with non-aggregated subsets of a transactional database. A load of raw data would typically result in a data model 80-90 percent of the size of the original data. V4 is an interpreted system with the basic operation being the evaluation of an intersection. Peak intersection rates on a 400+ megahertz Pentium-class processor can reach upwards of 150,000 intersections per CPU second. Performance is also based on the data and level of pre-aggregation. Scan rates of flat data have been clocked at over 50,000 rows per CPU-second, or 3 million rows per minute.

#### ***Summary***

Contextual database systems are an important new concept in information systems. V4 is currently the only package offering both multidimensionality and contextual evaluation. Only V4 can offer all of the following:

- Manipulation of concepts and points, not relations and rows;
- Representation of rules and facts in a single framework;
- Ability to store low-level transactional data and master file data as well as aggregated data;
- Incorporation of calculations/procedures as points;
- Use of context to allow different meanings to the same “words”;
- Use of context for sophisticated “what-if” analysis;
- Modeling and representation of metadata within the same framework as the target data;
- Ability to automatically generate low level queries from high level requests.