



ViloX™

Search On The Fly™ Technology

An Overview

By
Christopher M. Langan

Table of Contents

I. Introduction: Ad hoc Query Engines and Interfaces	3
II. Toward a Real-Time Database System	6
III. OLTP versus OLAP	10
IV. Recognition Over Recall	11
V. The Generality of the SOF™ User Interface	12
VI. SOF™ Analytics	15
VII. Applications	17
VIII. Economic Benefits	18

I. Introduction: Ad Hoc Query Engines and Interfaces

It has often been remarked that in the world of business, time is money. This is also true of information, *and is becoming truer by the day*. Unless enterprise personnel can get the information they need precisely when they need it, the enterprise suffers a net loss of efficiency, and this loss is ultimately reflected in its bottom line.

Modern enterprises store most of their information in *databases*, automated information repositories from which it must be extracted by or for people who use it to perform functions and make decisions affecting the enterprise. In practice, "for" is more common than "by"; information considered essential to routine operations is predefined, extracted at regular intervals, and distributed by IT departments to the personnel who require it. Unfortunately, the most critical events and circumstances tend to be precisely those that are *not* routine, and it is generally impossible to foresee when they will occur or what information will be necessary to respond to them.

Such exigencies require that decisions be made spontaneously, leaving no time to petition IT departments for relevant data. Responsible non-IT personnel thus require the ability to access mission-critical information themselves, in real time. Because corporations depend on information, they depend heavily on their IT departments; to some extent, this is (and will remain) unavoidable. But for all their technical expertise, IT personnel are only human and can only do so much, so fast. For this reason, mission-critical information held in databases should be accessible not only to those in IT, but to its end users on a personalized, up-to-the-moment basis.

Because enterprises are required to react promptly to unforeseeable circumstances, enterprise personnel must often make rapid decisions on the basis of unpredictable information. Where an *ad hoc query* is an impromptu request for information from a database, this means that the personnel in question must formulate and submit ad hoc queries to various databases. To do this, they must employ a *query engine* controlled through a *user interface*. Upon receiving an ad hoc query generated through the user interface and relayed over a network, the query engine delivers the query to the database for execution, collects the response, and then delivers the requested information back to the interface. The interface then displays the requested information to the user as a set of database records, i.e. as a *recordset*, or as a literal or graphical representation or numerical aggregate thereof.

The design and location of the query engine is important. It should neither drain the computational resources of the client nor compromise the security and efficiency of the database servers themselves. In addition, it must not place an unnecessary strain on bandwidth or be located where bandwidth limitations exist or would be likely to arise. This means that it must be designed to perform as many bandwidth-intensive transactions as possible without involving its clients, and that it be located near, but not necessarily on, the database servers. To meet this criterion, it must work in concert with the interface

and DBMS; while handling all of the necessary machine-level exchanges between itself and the DBMS, it must hold exchanges with the interface and its human user to a bare minimum.

Regarding the user interface itself, there are several methods of submitting queries to databases. Traditionally, the most powerful, complex and esoteric of these methods have directly employed a *query language* to communicate with the *database management system* (DBMS), a set of programs residing on the database servers and designed to store, modify and extract database information. While modern query languages are powerful, and in fact a marked improvement over their predecessors, they leave much to be desired when it comes to organizational efficiency and the maximization of human resources.

It is not hard to see why. In order for a DBMS to recognize and respond to real-world questions about data, the questions must be translated into a form intermediate between real-world human language and the computational language of the DBMS itself. Because a database is essentially one global relationship consisting of many lesser relationships, the structure of this intermediate "query language" mirrors *relational algebra*, a mathematical system describing the logical structure and dynamics of relations. Inconveniently, this renders query languages complex and cumbersome enough to require lengthy and expensive technical training for the attainment of fluency. A query engine that requires the user to type formal query language expressions into its user interface is thus seriously limited from an efficiency standpoint.

At the opposite end of the spectrum lies the easiest and (usually) the fastest method, choosing items from menus. By choosing a relevant descriptive parameter from a menu, the user defines a relevant subset of records in the database, prompting the DBMS to respond by presenting a new list of options appropriate to that subset. Unfortunately, such menus are usually designed in ways that limit users to certain sequences of choices and thereby skew the relational algebra of the database. As a consequence, flexibility, accessibility and efficiency are sacrificed, and the advantages of speed and simplicity are canceled by the disadvantages of inefficiency and lost information. That is, functional limitations built into the interface double as limitations not only on query compilation, but on the amount of useful information that can be accessed.

This leads to an obvious question: with respect to ad hoc data access, does there exist an intermediate method and interface combining power and ease of use?

In the most common intermediate method, *query by example*, the database program displays a blank record with a space for each *field*, or record attribute. The user then enters conditions for each field in order to define the query. For example, to find all products with prices greater than \$1000, one would enter *>1000* in the PRICE field of a blank product record. If one knows the meaning of symbols like ">", this method indeed qualifies as simple. In fact, it has been widely employed throughout the BI industry.

But while QBE systems are usually easier to learn than formal query languages, they tend to have severe drawbacks. For example, all of the fields may have to be filled in at once,

in which case the user must be able to fully formulate the query before submitting it. The user must know the value of each field without cues, prompts or subtle jogs of memory, and often without being able to see or experiment with the ways in which fields can affect each other through restrictions placed on them by specific value combinations in other fields. Unfortunately, the fallibility of human memory frequently renders these requirements unsatisfiable, and QBE unreliable.

And the problems do not end there. Also important is the relationship between the module that delivers the recordset and any modules that will be used to perform further operations or conversions on it...for example, modules that will be used to clean it or convert it into a report. As things have stood until recently, there has always been some measure of discontinuity between the process of locating the desired records and that of performing necessary operations on the recordset, up to and including messy and inconvenient changes of interface and module. Ideally, the juncture between record location and functional implementation would instead be as smooth and effortless as possible.

The first step towards solving these problems is to recognize that the recordset and its corresponding query are central to utility in all cases, and that the process of locating a desired recordset breaks down into two phases, *query design* and *query execution*. The goal is optimal synergy between these phases, and once the desired information has been delivered, seamless transition between the query stage and all subsequent functionality. That is, the object is first to make the design and execution phases of the query process work together, and then to seamlessly transit from the query process to any subsequent user-controlled operation on the resultant information. All user operations on a given recordset should be routed directly through the query interface in which it resides and through which it is located.

The ideal here being applied to database systems, that of putting the automated side of the computational loop in sync with the real-world side while maximizing efficiency at both ends, is called "real-time computing". From a narrow technical perspective, the prevailing wisdom is that no such thing exists with respect to human-database interaction...that although the magic phrase "real time" is often invoked by BI providers to emphasize the relative speed and/or convenience of their products, this constitutes a wishful misuse of the term. But in the long run, the narrow perspective seldom turns out to be the *right* perspective. The problem does have a solution. Query compilation and execution can indeed be synergized, and the means of locating a recordset can double as the means by which useful operations are performed on it.

In other words, real-time interactions between humans and databases are possible. But to understand why, we must acquaint ourselves with a little background.

II. Toward a Real Time Database System

The question of what constitutes *real-time* is controversial, and the associated terminology unstandardized. Of the several available technical definitions of the adjective "real-time", most apply specifically to fully automated systems in which there are no humans in the loop, e.g. automatic sensor-controller systems and digital signal processors. To apply the real-time concept to an enterprise, these definitions must be extended to systems containing both people and machines. And to apply the concept to systems involving human interaction with databases, they must be extended to fast, flexible systems in which people seek, locate, retrieve, manipulate, exchange and act on database information while leaving as much of the actual data-crunching as possible to Computers and software.

Standard real-time criteria include:

Time-sensitivity/criticality: A real-time system is one in which the value of an operation depends not only on its logical correctness, but also upon its time of completion. The system fails when its time constraints are not met.

Regularity (Predictability): A real-time system must be sufficiently predictable to guarantee the satisfaction of its time constraints.

Strict upper-bounding: A real-time system is one for which there exists an upper performance bound within the limits of its time constraints.

Realistic Load Capacity: A real-time system is one whose time constraints are satisfied under the most extreme foreseeable conditions of utilization, up to maximum user load.

These are the general criteria that must be met by any system claiming to be "real-time". Because of their hard-edged nature, the term is not usually applied to systems involving human beings with their relatively slow neurobiological reaction times; in the context of such systems, "real-time" is most often used as a synonym for "fast". But even in the context of human-machine information systems, real-time means more than this; it means that the system guarantees the delivery of critical information before its value is lost or compromised.

To understand what this means in the context of extended database systems in which people interact with hardware using software designed to extract information from data, some explanation is required. The timing constraints of real-time systems are just those of the enterprises and organizations in which they function. Because successful enterprises meet all of the above criteria, they are themselves the ultimate real-time systems. All of their subsystems, with or without humans in the loop, must function efficiently enough to permit the enterprise to act and react to internal and external conditions as quickly and appropriately as success may require. There should be no

critical internal inefficiencies; the only limitations that should exist are those due to external criteria beyond the control of the enterprise.

As critical components of enterprises, database systems are themselves real-time systems that must meet real-time criteria. Like the enterprises and organizations they serve, database systems have three types of components linked in a relationship of mutual necessity and facilitation: *users*, *software* and *hardware* (along with a fourth component, data, which constitutes the resource to be exploited). Because human utility is ultimately paramount, people are the central components of this relationship. Due to fundamental differences between human cognition and digital computation, they are also its most "problematic" components from an engineering viewpoint (just *as* digital devices are problematic from the viewpoint of cognitive neuropsychology).

While human psychology - or more precisely, cognitive language - is a function of the basic parameters of biological neural networks and human brains in particular, digital devices operate in terms of machine language, and we presently have no choice but to make the best of the rather extreme differences between these two kinds of language. It follows that any Solution for what an engineer might see as the "people problem" must be located in those components that directly relate digital computation to human cognition, and which thus lie "between" them in a functional sense. Concisely, where an enterprise functions by means of people making decisions and acting on information stored in databases, everything that happens at the human-machine interface should serve to minimize functional constraints on the human side, freeing people to do their jobs.

In the context of database systems, the above real-time criteria can be revised as follows:

Time-sensitivity/criticality: A real-time database system is one in which the value of data access, location and retrieval depends not only on logical correctness and relevance to global priorities, but also on timing. The system fails when its time constraints are not met.

Regularity (Predictability): A real-time database system must be sufficiently predictable with respect to data access, location and retrieval to insure that the satisfaction of enterprise boundary constraints will not be unnecessarily hindered.

Strict Upper-Bounding: A real-time database system is one for which there exists an upper bound on data access, location and retrieval within the limits of its time constraints.

Realistic Load Capacity: A real-time database system is one on which time constraints are satisfied under the most extreme conditions of utilization, up to maximum user load.

In other words, where a database system is an enterprise subsystem characterized by purposive interaction among multiple components including data, hardware, software, and human beings functioning on behalf of the enterprise, a *real-time* database system is one in which (1) each component of the system is individually adequate to meet system time constraints if properly configured with respect to the other available components, and (2) none of the components is improperly or inefficiently configured to such an extent that global time constraints are not satisfied. Thus, a real-time database system is one in which people are free to exploit their data resources "in real time", unhindered by obstacles associated with hardware or software.

In order to evaluate the functionality of any one of the components of a systemic relationship, it is generally assumed that everything else performs optimally; this fixes the analytical focus on the component in question. In order to apply this convention to the analysis of the software components of human-database systems, it must be temporarily assumed that people and hardware are performing at peak capacity, and the focus tightened on how the software facilitates human and/or mechanical interactions. Focusing on how the software relates people with people, people with hardware, and hardware with hardware lets it be evaluated with an eye to optimizing these relationships.

This kind of analysis yields what might be called a "hardware-wetware utilization" criterion: the software must drive the hardware, and assist job-related human mentation and performance, at or above a critical level of efficiency with respect to the extraction of useful information from stored data. This suggests the following definition:

Definition 1: A *real-time database system* is one in which the fundamental differences between cognition and computation are adequately reconciled by software, and in which the query-response process is therefore not critically hampered by software limitations.

Database system software can be partitioned into a DBMS, the user interface, and the transfer software connecting the two. DBMS programs, including the offerings of companies like Microsoft and Oracle, have evolved to the point at which they can reasonably be called "real-time" with respect to updates and query responses. However, there is much more to the software aspect of an extended database system than just the DBMS. Although database technicians are trained to communicate with a DBMS on its own terms "in real time", they (and it) generally operate behind the scenes in a supporting capacity with respect to other personnel who are not so-trained, and IT intervention is unavoidably associated with delays and other problems. To unlock the potential of the DBMS, all personnel must be allowed to interact with it to maximum efficiency.

In an extended database system, the DBMS is located with the database itself; in systems incorporating servers, database and DBMS generally reside on one or more servers. The user interface, on the other hand, is located where it can be accessed by users, e.g. on office computers, workstations and mobile and remote devices, and the associated query engine is optimally situated in the network with respect to information retrieval from these locations. This leads to the following refinement:

Definition 2: A *real-time query engine* is one by means of which the location and retrieval of available mission-critical information are not critically slowed by the design of the interface or functional constraints of the query engine associated with its structure or location, and in which dependent aspects of organizational performance do not suffer as a result of such problems. The only obstacles to the satisfaction of enterprise time constraints are the fundamental limitations of node and network hardware and

architecture, the DBMS, the data residing in the associated database, and the humans in the loop; no timing failure of the system is due to limitations of the query engine or its interface.

Some general observations are now in order. First, real-time constraints come in two varieties, those residing *inside* and those residing *outside the* systemic boundary. The former are largely controllable; the latter, which usually dominate, are not. Internal standards need merely suffice to meet external standards of organizational performance at the systemic boundary, and the practical meaning of *real-time* is determined accordingly.

Different enterprises have different boundary constraints. Those of a medical emergency services provider, where human life hangs in the balance, and even those of a shipping company, for which every minute a freighter waits in port can represent a huge, unrecoverable financial drain, may be different from those of an art supply company, where backorders are common and the pace is leisurely. These differences are naturally reflected in the practical meaning of "real-time" in various contexts.

The boundaries of enterprise systems are where performance and expectation meet. As technology evolves, so do expectations, and so do standards of competition and success in any given industry. What passes for "real-time" in a given industry in one economic setting may not pass in later ones; to define *real-time* on organizational success is to define it on the satisfaction of expectations, and as technology evolves, expectations rise with respect to organizational performance. Accordingly, the phrase "real-time" actually refers to an evolutionary optimization process largely driven by expectative boundary conditions, and standards of rate optimization must be general and/or flexible enough to apply throughout this evolutionary process.

In fact, the entire evolution-optimization relationship is problematic. Just as optimization drives evolution, evolution can redefine optimization. Because the process of optimizing system components is parameterized by properties of the system and its other components, the means and standards of optimization applying to any subset of system components evolve with the system. Thus, "absolute (real-time) optimality" can be defined only with respect to those features that are invariant with respect to systemic evolution. Where not explicitly qualified in this way, optimality is implicitly relativized to current levels of expectation and prevailing organizational and technological paradigms.

In other words, *real-time* is, and will remain, a tall order. Even if it is not immediately clear how to determine whether or not an interface and query engine contain everything that might speed up the design and execution of every possible kind of ad hoc query now and forever, it is clear that if performance fails to meet expectations at any juncture, the price to a real-time enterprise can be both exorbitant and payable on the barrelhead.

Accordingly, in order to allow for changing conditions and expectations, and attendant improvements, extensions and updates of real-time technology, the definition of *real-time* must be applied to key features of general methodology rather than to any particular

embodiment thereof. However, as will be explained below, it is safe to assert that on methodological grounds, patent-pending SOF™ Technology by ViloX™ constitutes so major an advance over the rest of the field that no BI provider except ViloX™ can credibly claim optimality for its own ad hoc query engine or interface.

III. OLTP versus OLAP

Databases may be generally categorized by their *schemata*, or the ways in which data are related within them. There are two main types of database schema, *relational* and *star*, and two corresponding types of database processing, OLTP (*On-Line Transactional Processing*) and OLAP (*On-Line Analytical Processing*). Relational databases are designed for the moment-by-moment storage and retrieval of current data; OLAP databases are designed for multilevel aggregation of data into (e.g.) time series. For example, a user might use OLTP to query a relational database for an up-to-the-minute list of transactions made today through a given outlet, but use OLAP to query a star database for information on total weekly sales for each store in the Eastern Division for each week over the last year.

Note that both OLAP and OLTP contain the phrase "on-line", for which *real-time* is sometimes employed as a synonym. By this standard, both OLTP and OLAP could lay claim to being "real-time" processes. However, this is clearly misleading, since OLAP requires extra time for data aggregation that OLTP does not require. In fact, the aggregative time requirements of OLAP are often on the order of hours or even days, and this is flatly inconsistent with the definition of *real-time*. By the time a piece of data gets to the OLAP aggregation stage, its currency has lapsed.

It follows that if there actually exists such a thing as a "real-time database", its structure cannot be optimized for OLAP, which is itself designed to deal with extended time series and not with current data. Quite simply, a "real-time OLAP database" is an oxymoron. OLAP databases can contribute to the success of an organization, but over broad strategic timescales rather than moment by moment. Thus, the stress currently placed on OLA~ by most BI providers conduces to the strategic rather than the ad hoc tactical level of planning. Because the tactical level is the real-time level, OLAP is antithetical to the real-time imperative.

What about real-time OLTP? While OLTP is in principle a true real-time process in the sense given above, it is subject to certain rate limiting factors and performance bottlenecks. Where the goal is to avoid unnecessarily restricting the efficiency of humans in the loop, the most important of these bottlenecks involves the user interface. Where the interface is suboptimal, it can cause enough of a delay to interfere with human performance.

What properties must a real-time OLTP interface possess? Three related features are especially important. First, the interface must accommodate *user logic succession*; because human thought patterns are unique to those who generate them, it must not restrict the user to any predefined sequence of choices. Second, due to the fact that the human mind works by association on sensory cues, the interface must provide a maximum number of cues to the user at each step of the query design process; the contents of recordsets should be laid out as revealingly as possible. And third, the interface cannot require the complete construction of queries prior to execution. The construction and execution of queries should be as synergistic and concurrent as possible; as soon as a segment of the query is decided, it should be executed immediately so as to provide associative cues and thereby aid in the next stage of construction.

Together, these key criteria add up to **Recognition over Recall**.

IV. Recognition Over Recall

Almost all information processors, including digital computers and human minds, are capable of recognition and memory. However, these functions work very differently in brains and computers. The recognition function of the brain has high capacity for association and generalization, while that of a computer is extremely narrow and literal. On the other hand, the memory function of the brain is selective and often unreliable in both storage and retrieval, while a computer's ability to store and retrieve specific details is downright amazing. In other words, computers have an exhaustive memory for details, while the brain, which needs only minimal cues to generate an entire network of associations, excels at recognition in low-information environments.

Since efficiency means delegating specific tasks to the processor types that do them best, efficiency in querying means delegating recognition to human brains, and fact retrieval to computers and databases. This means dedicating the human end of the query process to the recognition of likely attributes, and the machine end to the retrieval of specific details and other information to which human powers of association and intuitive decision-making may be applied. This leads to a general design principle: *the user interface of an), query application should accentuate human powers of association and leave exact, exhaustive recollection to the computer and DBMS*. ViloX™ calls this GUI design principle "Recognition Over Recall".

In any proper implementation of this principle, the query interface provides the user with a recognizable, e.g. alphanumeric, classification of selections over a full range of options. Taking in all of the available selections at a glance, the user can then choose the one that stimulates the most promising mental associations, preferably with a single mouse click. This click should exploit the data-crunching power of the computer to rapidly produce a new, narrower range of possible selections, and so on until the desired information has

been located. Finally, again with a single mouse click, the user should be able to download, print, or apply any available function to the data thereby located. There should be no wasted information and no wasted effort.

At the present time, ViloX™ is the only company that successfully applies this principle. The vehicle of implementation is a revolutionary ad hoc query engine called *Search on the Fly™* or *SOF™*. The SOF™ implementation of Recognition Over Recall makes the design and execution phases of the query process work together; each time a relational criterion is selected from a compact alphanumeric list of exhaustively disjunctive possibilities, that piece of the query is automatically executed, and the corresponding results are returned to provide a basis for further refinement of the query until the desired information is located. Queries are thus synergistically constructed and executed in a stepwise fashion, a new piece of the puzzle being fit into place and a new set of clues about the next piece appearing whenever a menu item is clicked.

At this point, it should be obvious that the SOF™ implementation of Recognition Over Recall alleviates the ubiquitous efficiency loss associated with complex ad hoc queries and joins by simply eliminating the problem. The problem does not arise because there is no such thing as a complex ad hoc query or join in SOF™. As the user issues point-and-click commands according to his or her personal pattern of recollection and association, the query is cumulatively executed in the very process of design and construction.

There are excellent reasons to believe that the SOF™ method of data access is optimal with respect to current computational, informational and network architectures. Never have mind and machine, client and server, or data access and data manipulation been made to work so closely and successfully with each other in the context of extended database systems.

V. The Generality of the SOF™ User Interface

As mentioned in the first section of this document, a relational database is essentially just a logical relationship containing lesser *relationships* among its own records. Together with the basic logical operations that can be performed on its data to extract relational content, it comprises a *relational algebra*. The relation concept is absolutely general; since every piece of information is by definition a relation, every bit of usable information in a database can be extracted using relational algebra.

To elaborate, an *algebra* is an abstract structure consisting of sets of elements and operations thereon. A relational algebra, being an algebra designed for manipulating relations, takes sets of logically related records as its operands. Owing to the operational equivalence of sets and relations, the operations of this algebra include the usual set-theoretic operations *union*, *intersection*, *difference* and *Cartesian product*, along with

several additional operations involving attributes, including *selection*, *projection* and *join*. In addition, there are several other operations not usually considered fundamental, including *division*, *renaming*, *assignment* and *composition* (of relational operations). Yet other operations can be defined.

In the relational context, the set-theoretic operations are defined as follows. Where R_1 and R_2 are relations, union ($R_1 \cup R_2$) is the relation containing all records that appear in R_1 , R_2 or both; intersection ($R_1 \cap R_2$) is the relation containing all records that appear in both R_1 and R_2 ; and difference ($R_1 - R_2$) is the relation containing all records of R_1 that do not appear in R_2 . In order to apply these operations to relations, operands and their products must share the same attributes. Finally, Cartesian product is the relation consisting of all possible pairs of records from a pair of relations

Defining the select operation requires the concept of a *predicate*. In the present context, a predicate is a Boolean (2-valued, true-or-false) expression whose operators consist of the logical connectives *and*, *or*, and *not*, and the arithmetical comparisons *less than* ($<$), *less than or equal to* (\leq), *greater than* ($>$), *greater than or equal to* (\geq), *equals* ($=$) and *not-equals* (\neq), and whose operands are either domain names or domain constants. The select operation can now be described as identifying within a given relation all of the records whose attributes satisfy (make true) a predicate corresponding to the selection criteria. E.g., $R_2 = \text{select}(R_1, P)$ creates a new relation R_2 by selecting from R_1 all of the records that satisfy the predicate P .

Just as selection pulls records from a relation, projection pulls attributes. Concisely, the projection operation distinguishes a subset of the attributes in a relation and discards the rest. E.g., $R_2 = \text{project}(R_1, F_1, \dots, F_n)$ creates from the records in R_1 a new relation R_2 containing only the attributes F_1 through F_n . The join operation, on the other hand, combines two relations to form a third relation containing *all* of their combined attributes. I.e., $R_3 = \text{join}(R_1, F_1, R_2, F_2)$ creates a combined relation R_3 with all of the attributes of R_1 and R_2 by scanning all possible pairs of records from R_1 and R_2 respectively, and for every pair with equal values in the chosen attributes F_1 and F_2 , adding to the result a record containing all the attributes of both R_1 and R_2 and merging the duplicate fields F_1 and F_2 . Where R_1 and R_2 share just one attribute, it is assumed that this is the one to be compared for equality, and the join is said to be "natural"; otherwise, it is of another variety corresponding to its relational criteria.

Of the above operations, only four or five are usually considered fundamental: union, select, project, and some combination of other operations including join, renaming, difference and Cartesian product. Whichever operations are used as a basis, all of the others can be expressed in terms of them. For example, where difference is considered fundamental, intersection can be expressed in terms of difference: $P \cap S = P - (P - S)$, and where Cartesian product is considered fundamental, the join operation can be regarded as a select operation performed on a Cartesian product. Regardless of how it is chosen, a full operational basis affords access to every bit of information a database contains. But no matter how few and simple the chosen basis may be, putting its operators together in such a way as to find a particular piece of information can be difficult.

Because Standard Query Language incorporates relational algebra in its syntactic structure, the same is true of SQL *a fortiori*. Using SQL, it is possible to formulate machine instructions for performing all of the operations of relational algebra, and thus for retrieving any or all of the relations in a given database. The power of SQL can thus be said to reside in its versatility. However, as already remarked, the primary weaknesses of SQL are the same as those of the relational algebra it mirrors: complexity and a very steep learning curve. Of the many enterprise personnel who require direct access to database information in order to do their jobs efficiently, only those in IT departments are likely to possess the training and experience necessary to use SQL effectively,

Until recently, this impasse rendered the phrases *database* and *IT department* synonymous with *information bottleneck*. Although a number of BI providers responded by presenting supposed "solutions" in the form of alternative methods of database access, their offerings were without exception too weak and/or cumbersome to save the day. The limitations associated with prefabricated menus and query-by-example techniques were simply too restrictive, and more powerful methods remained unintuitive, time-consuming, and too hard to learn.

It was at this point that ViloX™ entered the picture and solved the problem. The solution process can be described in a stepwise fashion.

1. Attacking the problem as a team, ViloX™ theoreticians and engineers characterized the general types of information required by enterprise personnel.
2. ViloX™ characterized the most natural and effective way in which human beings can interact with symbolic information in representationally-bounded environments: through intervalized, nominally-truncated alphanumeric lists.
3. ViloX™ identified a small handful of fundamental relational operations, *select*, *rotate*, *merge* and *negate*, possessing two key properties: (A) applied to alphanumeric lists, they are together capable of locating all of the required relations within any database, and (B) they can be dynamically configured to parallel the associative and inferential processes of the human mind.
4. Guided by a combination of computational and neurocognitive principles, ViloX™ subjected this parallel dynamic to several stages of ergonomic refinement and optimization. By providing direct inferential pathways, affording the right associative cues, and minimizing wasted effort by eliminating dead ends and providing backtrack capability, the number of steps required to locate virtually any piece of information in any database was minimized, and the learning curve virtually flattened.
5. ViloX™ devised a universal GUI (*Graphic User Interface*) through which any user, by intuitively applying the operations in question to alphanumeric lists in any desired order on any desired sequence of items, can find any desired type of information given only minimal knowledge or recollection of the operative selection criteria. The method of application was designed to be as simple as possible: a mouse or other pointing device,

including voice recognition over a minimal command vocabulary, would be used to define a layered sequence of predicate filters on lists presented in cascading menus or serial HTML pages. Keyboard use was thus limited to keyword entry. This allowed ViloX™ engineers to immediately adapt the interface to a wide range of computational and electronic devices including PDAs, cell phones, and television set-top boxes.

6. ViloX™ designed a query engine based on the interface. This engine was designed to work with any number of databases running on any number of platforms, including legacy platforms. In keeping with its real-time character, it was also designed for minimum maintenance and maximum centralization and ease of installation.

7. ViloX™ worked to optimize the communication between GUI and DBMS in various organizational, computational and network environments. This resulted in an optimal division of labor between user and DBMS, and a concomitant minimization and optimal distribution of bandwidth requirements.

8. ViloX™ amplified the GUI by providing iconic access to numerous user-controlled operations on located information, including numerical aggregation, the fast design and generation of reports, and data cleansing through the consolidation of similar but differently-named, independently-stored records. Once a set of records has been located using the SOF™ GUI, it can immediately be subjected to further operations from within the GUI itself. These operations correspond to icons at the tops of the menus.

9. ViloX™ developed several lines of complementary products leveraging the power of the interface and engine. These products perform various industrial-strength functions including ETL, schema extraction, and website control and optimization.

10. ViloX™ performed a variety of small-, mid- and large-scale enterprise installations, gradually toughening the software and evolving it to the pinnacle of reliability.

Having successfully passed through these stages of development, SOF™ real-time technology can presently be described as mature, robust and dependable.

VI. SOF™ Analytics

Infoconnectivity~ (informational connectivity), defined as that property of organizational structure by virtue of which all of the members and departments of an organization can communicate with others to the extent required in order to meet organizational boundary criteria, divides analytics into two *categories: primary* and *secondary*. Primary analysis emphasizes the establishment of infoconnectivity and the empowerment of enterprise and IT personnel to quickly and independently locate specific pieces of information necessary in order to function from minute to minute, while secondary analysis places the emphasis

on *post hoc* statistical analyses of large swarms of data to the end of long-term planning and coordination, without respect to whether infoconnectivity has been established. The kind of analysis generally provided by mainstream BI developers is secondary by default, while that provided by ViloX™ is primary.

This distinction can be further explained in terms of decision (or game) theory. Like military planning and operations, enterprise operations can generally be stratified into *strategic and tactical* levels. The strategic level is obviously concerned with strategy or long-range planning, while the tactical level is associated with the on-the-fly, action-reaction dialectic of transactional event-response sequences. Analytics can thus be divided into strategic and tactical phases.

Mainstream BI (*business intelligence*), its real-time deficiencies having pushed it in the direction of OLAP and user interfaces based on fancy statistical display techniques, has overwhelmingly gravitated toward strategic analysis, all but abandoning the ad hoc, real-time tactical phase for want of adequate methodology. With respect to tactical analysis, the deficits of mainstream software remain as egregious as they are notorious. Corporate personnel forced to rely on mainstream BI report that the associated analytics, which are so thoroughly dominated by the inherent limitations of OLAP that the efficiency of query engine and interface are all but irrelevant, can literally take days to execute.

Because enterprises constantly generate data, enterprise databases are not static entities. They must be continually updated. Every time an OLAP database is updated, aggregates must be recomputed. OLAP requires that this be done exhaustively, every dimension being combined with every other in every possible way. This takes an amount of time that no real-time enterprise can reasonably afford.

However, the problem turns out to be entirely avoidable. Advance characterization of queries is what allows the design of a star schema and the exhaustive compilation of the corresponding aggregates in OLAP. But if queries can be characterized in advance, then they can just as easily be predefined to run cumulatively in an OLTP setting, keeping a running tally of key aggregates as the source databases evolve over time.

In other words, if one knows enough about the queries that will be run against an OLAP database to effectively design and maintain it, one can just as easily predefine the same queries to run against the original relational databases from which the OLAP database is periodically compiled. On the other hand, because predefined OLAP queries require repetitive exhaustive data compilation while OLTP queries do not, cumulative aggregates can be more selectively and efficiently obtained by means of OLTP.

This is the approach taken by ViloX™. In place of the customary OLAP warehousing of data, ViloX™ permits "snapshots" of essential aggregates to be periodically extracted and saved as time series. Without the overhead and maintenance of a full-blown OLAP warehouse, computational and financial resources can be more efficiently and gainfully utilized.

This being understood, it must now be noted that SOF™ technology is perfectly compatible with OLAP. SOF™ queries can be run against existing OLAP databases as easily as they can be run against a dynamic relational database. Accordingly, SOF™ is at the very least a valuable addition to any existing OLAP data warehouse, freeing the proprietary enterprise from OLAP limitations while facilitating true real-time OLTP.

VII. SOF™ Applications

Just as real-time SOF™ Technology allows enterprise personnel to bypass information bottlenecks associated with obstructive interfaces, inefficient software and overburdened IT departments, it also represents an unparalleled opportunity for IT departments to find relief of their own. In the hands of competent information technicians, the ability to summon and control industrial-level database applications directly through a truly effective query interface is nothing short of invaluable. With its easy maintenance requirements and an installation and adjust™ ent period of days or weeks rather than months, SOF™ has the potential to elevate any data-intensive enterprise to new levels of efficiency and productivity.

Among the IT-level applications within easy reach of ViloX™ technology are database schema extraction, industrial reporting, establishment and maintenance of web portals, general website control and optimization, and ETL (*Extract, Transform and Load*) functions involving the scheduled merging of select data from multiple databases into consolidated stores. Yet another is enabling one or more administrators, through centralized, easy-to-use command modules, to provide various departments with standardized SOF™ access to specialized data collections individually sculpted to the purposes and access levels of various personnel. This empowers an IT department to ensure and maintain the infoconnectivity of the enterprise, freeing its employees from IT time constraints and crippling dependence on IT intervention. By providing personnel with just the required levels of access to data resources, a ViloX™ -equipped IT department can create a securely compartmentalized but easily-navigated data environment, and thus provide the enterprise with a stable, efficient informational infrastructure.

Other industries in special need of real-time database technology include banking, credit and accounting; finance and portfolio management; the travel and hospitality industries; the shipping and delivery industries; all aspects of the insurance industry; the real estate industry (brokerage, acquisition and maintenance); all aspects of medical services, supply and administration; manufacturing, wholesale and retail products and services, including real-time CRM; web-based e-commerce (where anything less than immediate access to desired products and services leads to immediate customer attrition); corporate security and law enforcement; bioinformatics and pharmaceutical research; other forms of scientific, financial, actuarial and forensic research;

corporate administration, and all other fields in which people perform functions requiring fast, direct access to stored data containing relevant information.

In sum, by virtue of its generality, versatility and power, Search On The Fly TM Technology by ViloXTM qualifies as the long-awaited embodiment of what once seemed an unattainable ideal: *real-time database computing*, including dynamic user-database interaction and data access.

VIII. Economic Benefits

Due to international political uncertainty and increased volatility in the high technology sector of the world market, corporations now face economic conditions that are prone to more or less violent and unpredictable fluctuations. This is forcing many companies to stop relying on corporate inertia, and look for ways to improve efficiency, reduce overhead, and streamline their operations.

The *20-80 rule* is a pricing rule of thumb in the field of industrial software. It states that out of the total cost of a major industrial software installation, 20% pays for the software and 80% pays for installation and maintenance. With respect to this rule, ViloXTM SOFTM technology generally offers a dramatic improvement. Along with other economic advantages of SOFTM technology, the reasons are listed below.

1. Where human beings are expected to make decisions and/or act on the basis of current information, human minds are costly things to waste. By putting a truly effective means of data access, retrieval and manipulation directly in the hands of those who use the data, SOFTM technology cuts to a bare minimum the waste of human mental resources with respect to making well-informed *ad hoc* decisions.
2. In addition to effecting an optimal division of query labor across the man-machine divide, the SOFTM engine distributes data traffic in such a way that wasted man-hours are reclaimed and network efficiency is vastly improved. Accelerating the flow of information means accelerating the flow of business, and accelerating the flow of business means accelerating the flow of income.
3. SOFTM has an ultra-thin client that can run on virtually any modem digital device, relaying requests and responses between client and server with minimum bits at maximum sequential transmission rates. By making the most of computational resources on both sides of the query interface, SOFTM technology makes it possible to extract maximum value from current hardware capitalization. Companies whose data access problems would once have forced expensive hardware upgrades can now delay such expenditures by using SOFTM to make the best of their current resources.

4. Shorter installation periods mean lower installation costs. Compared to installations of other types of BI software, S OF installations are typically fast, simple and painless. By replacing many installation technicians with a few, and limiting the term of their involvement to days or weeks instead of months, ViloX™ cuts the large initial outlays traditionally associated with major software installations to a fraction of the norm.
5. The ruggedness and simplicity of SOF™ technology can radically lower maintenance costs. Companies doing business with mainstream BI providers must pay for what often amount to full-time support teams in order to work through the bugs and breakdowns that inevitably occur when BI software is overly complex and/or fundamentally inadequate to satisfy real-time criteria. With SOF™, this is not the case; a short break-in period usually suffices to iron out any relatively minor problems that might arise, and routine maintenance is minimal.
6. Whereas most kinds of BI software require that users receive considerable formal training, and thus entail considerable corporate expense, SOF™ is as simple, intuitive and easy to learn as possible. An initial group orientation is usually adequate to bring even the most technically illiterate personnel up to speed, and to allow an enterprise to immediately start extracting maximum value from its data resources.
7. Whereas increasing demand for data once required the hiring of additional IT personnel, SOF™ technology enables fewer IT personnel to satisfy the needs of more users while handling routine tasks with greater efficiency. SOF™ helps even downsized IT departments perform adequately with less manpower.
8. The radical new design of ViloX™ technology ensures that it will remain state-of-the-art. Due to its innate real-time functionality, it will not become outdated and thus necessitate a costly and disruptive switch to "something faster".
9. Thanks to the simplicity and directness of SOF™ technology, customers using it to search for products and/or services are not put off by unnecessary complexity, and are less likely to be driven by impatience and frustration to take their business elsewhere.
10. Because the majority of corporations still use outmoded BI software, ViloX™ technology has the potential to confer a profound competitive advantage on those who first adopt it...and the ability to *stay* competitive even as others follow suit.

So much for the advantages of ViloX™ products; what about the company itself? Due to the extreme power and simplicity of its products, ViloX™ has been able to remain streamlined and avoid the huge overhead of mainstream BI providers, making it more stable and less vulnerable to economic vicissitudes. Thus, ViloX™ is more likely than other companies to weather any impending storms, and figuratively speaking, to spare its customers any future expense and inconvenience associated with "changing horses in the middle of the data stream". As the demand for real-time data access inevitably grows, so will ViloX™. Unlike mainstream companies forced to rely on price exploitation of existing

customers, ViloX™ will be able to rely on the kind of steady market penetration that comes only with clear product superiority and customer satisfaction.

Over the last several years, the business world has been repeatedly alerted to the persistence of certain harsh economic realities that almost seemed to have faded out of existence before "the bubble burst". Today, ignoring these realities is no longer a viable option. For all of the above reasons and more, SOF™ Technology by ViloX™ is an eminently rational answer to the series of wake-up calls even now being heard by responsible corporate executives across the globe.

About the Author:

Christopher M. Langan

Bio - Is an [American](#) whom numerous media sources report as having an estimated [IQ](#) of 195, as reported by [20/20](#), [BBC](#), [Esquire](#), [Extra](#), [Fantástico](#), ["First Person"](#), [Inside Edition](#), [New York Newsday](#), [Popular Science](#), [The Times](#), and others. According to [20/20](#), Langan scored "off the charts" when tested by Dr. Robert Novelly. Novelly, a board certified [neuropsychologist](#), commented that Langan was "the highest individual that I have ever measured in 25 years" of testing. In [2001](#) Langan was featured in [Popular Science](#) magazine, where he discussed his "Cognitive-Theoretic Model of the Universe" (CTMU), a philosophical model of reality. Arguing that theories and inferences, including inductively-derived laws of nature, are bound together in a more general relationship between mind and reality, Langan explores the implications of this idea in various contexts including physics and cosmology, biological origins and evolution, psychology, ethics, and theology in a 56-page paper published in 2002. Langan's ideas on physical and biological causality were further explicated in Chapter 13 of *Uncommon Dissent: Intellectuals Who Find Darwinism Unconvincing*, a collection of essays published by the Intercollegiate Studies Institute in 2004. Langan is a fellow of the [International Society for Complexity, Information and Design](#), a think-tank founded by leaders of the [intelligent design movement](#) that describes itself as a "cross-disciplinary professional society that investigates complex systems apart from external programmatic constraints like materialism, naturalism, or reductionism.". He also serves on the board of the [Mega Foundation](#), a nonprofit foundation for the [gifted](#)

ViloX, Inc.

231 Breckenridge Lane, Suite 201
Louisville, KY 40207
Telephone: 502-561-3472
Fax: 502-371-0651
Email: info@ViloX.com
Website: www.ViloX.com

(The terms ViloX, *Search On The Fly*, and *SOF* are trademarked by ViloX, LLC)

© 2008 by ViloX, LLC