Access to Simulations in Order to Extend Information Systems

Gio Wiederhold and Rushan Jiang
Stanford University
Computer Science Department.
Gates Computer Science Building 4A
Stanford CA 94305-9040
650 725-8363 fax 725-2588
<gio, jiang@db.stanford.edu>

Abstract

We have developed a prototype system to provide access to simulations in order to support the decision-making functions of information systems more completely. The central component is a new interface language, which we call SimQL, which mirrors the functionality of SQL, but provides information from a variety of simulations so that results can be projected into the future. Simulations to be wrapped for SimQL access range from spreadsheets to large remote continuous simulations, as used for weather forecasting. Results from SimQL produce pairs of data elements, the expected value and its certainty. SimQL is not a language for programming the supporting simulations, just as SQL is not the language used to write a DBMS. The motivating concept is that having an interface language allows separation of customers and information providers. In turn, the autonomy created by the language interface allows progress by information customers and their information providers to be made independently.

The intent of SimQL to support an architecture where information systems not only extend their capabilities into the future, but also support the assessment of the effects of alternate decisions, so that multiple future courses can be compared. We expect that making results of simulations as accessible as other information components, as databases and web-based data are today will greatly augment the effectiveness of integrated information systems for end-users.

Motivation

Basic database systems are being extended to encompass wider access and analysis capabilities. The objective of many extensions is to provide more capabilities for decision-making. However, the decision maker also has to plan and schedule actions beyond the current point-in-time. Databases make past and nearly current data available, but simulation tools are required for projecting the outcome at some future time of the decisions that can be made today. The tools that are available for projections range from back-of-the envelope estimates, include spreadsheets, business-specific simulations, to continuous simulations, as used for weather forecasting. These tools provide information which is complementary to the information about the past provided by databases, and help in selecting the best course-of-action [LindenG:92]. Examples are obvious when dealing with business decisions, resource allocation, crisis situations, and the like. The problem has been recognized in military planning; quoting from [McCall:96]: The two `Capabilities Requiring Military Investment in Information Technology' are:

`1. Highly robust real-time software modules for data fusion and integration;
`2. Integration of simulation software into military information systems.'

Today rapid progress is being made in information fusion from heterogeneous resources such as databases, text, and semi-structured information bases [WiederholdG:97]. Results of this research are being transferred to practical settings. However, the predictive requirements for decision-making have been rarely addressed in terms of integration and fusion [Orsborn:94]. The most common tool being used for planning and documenting predictions is the spreadsheet. In situations where the amount of data is modest and relatively static, files associated with

spreadsheets have supplanted the use of database technology, and extensions to allow sharing of such files are being developed [FullerMP:93]. Our research is intended to support such tools within the database paradigm.

Infrastructure:

Technology has made great strides in accessing information about past events, stored in databases, object-bases, or the World-Wide Web. Data warehouses that integrate data into historic views are becoming broadly available [Widom:97]. Access to information about current events is also improving dramatically, with real-time news feeds and on-line cash registers. Extensions to SQL to manage historical data are becoming well accepted [Snodgrass:95] . We must still expand the temporal range to project the effect of candidate events into the future, and manage the uncertainty of such projections.

The importance of rapid, ad hoc, access to data for planning is well understood, but should not be limited to historic data. Decision-making in planning depends on knowing past, present, and likely future situations, as depicted in Figure 1. To assess the future computationally we must access simulations. Simulations employ a wide variety of technologies, including continuous equational models and discrete, time-step models. Many simulations are available from remote sites [FishwickH:98]. Simulation access by more general information systems should handle local, remote, and distributed simulation services. Distributed simulations can also communicate with each other [MillerT:95]. They interact using highly interactive protocols (HLA) [IEEE:98], but their results are not accessible to general information systems [Singhal:96]. If the simulation is a federated distributed simulation, as envisaged by the HLA protocol, then one federation member may supply the data to the decision-making support system, by first aggregating data from detailed events to the level that is appropriate for initiating planning interactions.

Already, when historic records are a bit out-of-date, planners routinely make undocumented projections to extrapolate from a past, known state to the current situation in order to obtain an approximate current picture. Extrapolating further into the future increases the uncertainty. Uncertainty is an essential aspect of planning, and has been studied in a variety of abstract settings [BhatnagarK:86]. It will be important to provide a linkage to this research in practical, information-based planning systems.

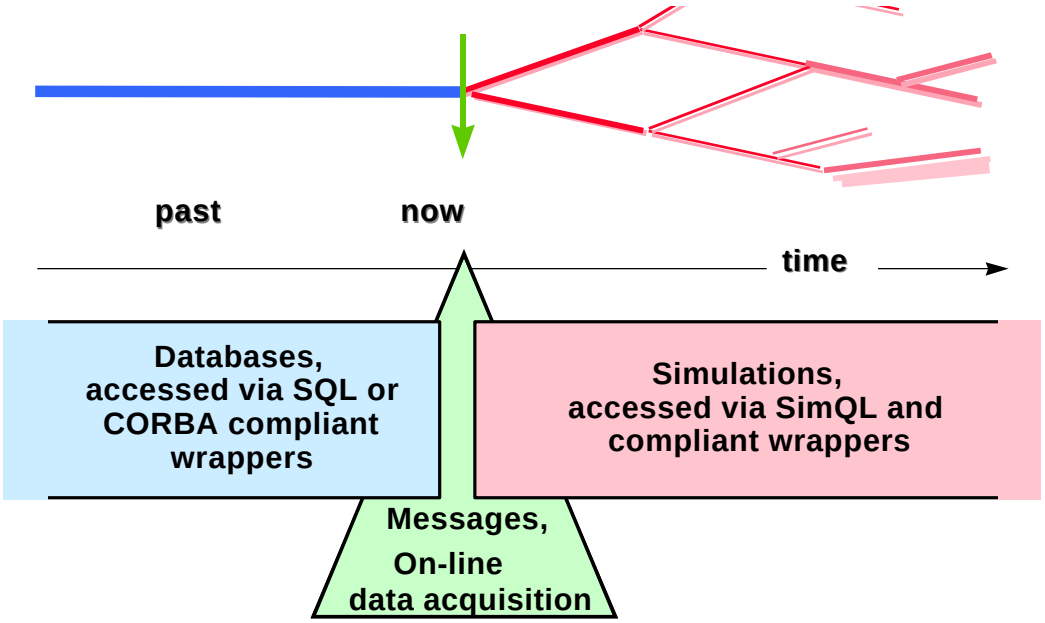

Figure 1: The Place of Simulation Access in Information Systems

Concepts:

The concept of our simulation access language, SimQL, mirrors that of SQL for databases. Modern versions of SQL provide now also remote access [DateD:93]. An ability to access simulations as part of an information system adds a significant new capability, by allowing access to factual data and to projections of their values into the future, given that certain action occur. Actions may be specified by the client as input, say a decision to use air freight rather than road transport, or may come from other simulations, say a snowstorm causing delays in Chicago. Interfaces languages such as SimQL should be able to exploit emerging conventions for information systems. For instance, they might use a CORBA communication framework, and `Java' for client-based services. Integration of SimQL access interfaces within larger client systems distinguishes our work from the objective of building grander simulations, which motivates the simulation community.

To make the results obtained from a simulation clear and useful for the decision maker the interface must use a simple model. Computer screens today focus on providing a desktop image, with cut and paste capability, while relational databases use tables to present their contents, and spreadsheets use a matrix. To be effectively used simulations should also present a coherent interface model. Since we expect to often have to integrate past information form databases with simulation results we start with the relational model. However the objects to be described have a time dimension and also an uncertainty associated with them. We hence will use a simple object model as the descriptive interface for SimQL.

Tools for Prediction

Projecting the outcome of current decisions into the future requires some form of simulation. Often such a simulation is only performed in a planner's mind. That is, the planner sketches reasonable scenarios, mentally developing alternate courses-of-action, focusing on those that had been worked out in earlier situations. Such mental models are the basis for most decisions, and only fail when the factors are complex, or the planning horizon is long. Human short-term memory can only manage about 7 factors at a time [Miller:56]. That matches the number of reasonable choices in a chess game, so that a game as chess can be played about as well by a trained chess master as by a dumb computer with a lot a memory. But chess is simpler than most of the real world. When situations become more complex, tools for pruning the space of alternatives, presentation of viable choices, and their comparison become essential . Today, the pruning is mainly done intuitively, the presentations use whiteboards. Available tools are video-conferences and communicating *smartboards,* sometimes augmented by pasting results that participants extract from isolated analysis programs. For instance, a participant may execute a simulation to see how a proposal would impact people and supply resources. Financial planners will use spreadsheets to work out alternate budgets, and show a subset of the parameters to others.

When the simulations incorporate alternate assumptions, they will produce alternate futures, so that an information model that supports planning must not only incorporate uncertainty, but also further alternatives, as shown in Figure 1. Events outside of the decision-makers control, as responses by others or acts-of-nature, lead to subsequent alternate future scenarios. At each ply the alternatives multiply, and pruning or coalescing becomes crucial. As time passes, opportunities for choosing alternatives disappear, so that the future tree is continuously chopped off at the root as the *now* marker marches forward [CliffordEa:97]. Also, expected values may change and their uncertainties reduce as time passes , so that the tools that provided the information about the future will have to be re-invoked. Keeping information models for planning up-to-date is hence much work, and is unlikely to happen without tools that automate the integration of information about the future into decision-support systems. Our work in SimQL provides the interfaces for such tools, but did not extend to implementations of the multi-branch information systems that motivate our research.

The SimQL Approach

SimQL provides an interface for accessing information about future events. We focus on accessing pre-existing predictive tools. Wrappers are used to provide compatible, robust, and `machine-friendly' access to their model parameters and execution results [HammerEa:97]. Our wrappers also convert the uncertainty associated with simulation results (say, 50% probability of rain) to a standard range (0.5 out of 1.0 -- 0.0). If the simulation does not provide a value for its uncertainty the wrapper may estimate a value based on experience of its author.

In terms of system structure, we follow the accepted SQL approach. Consider that SQL is not a language in which to write a database system; those may be written in C, Ada, etc.. Rather, SQL is a language to describe, select, and fetch results for further use in information systems. The databases themselves are owned and maintained by others, as domain specialists and database administrators. Similarly, use of SimQL enables access to the growing portfolio of simulation technology and predictive services maintained by others in the simulation community. Having a language interface will break the bottleneck now experienced when predictions are to be integrated with larger planning systems.

In particular, there are two aspects of SQL that SimQL mimics:

I. A Schema that describes the accessible content to an invoking program, its programmers, and its customers.

II. A Query Language that provides the actual access to information resources.

Using similar interface concepts simplifies the understanding of customers and also encourages seamless interoperation of SimQL with database tools in supporting advanced information systems. There are differences, of course, in accessing past data and computing information about the future:

1. Not all information about a simulation is made accessible via the SimQL schema. Simulations are often controlled by hundreds of variables, and mapping all of them into a schema for external access is inappropriate. Only those variables that are needed for querying results and for specifying the simulation ranges will be made externally accessible. The remainder will still be accessible to the simulation developer. Defining the appropriate schema requires the joint efforts of the developer, the model builder, and the customer.
2. Predictions always incorporate uncertainty. Thus, a measure of uncertainty is always reported with the results. There have been multiple definitions of uncertainty [BhatnagarK:86], but we have only worked with a single value. Its interpretation requires insights by the client programmer, just as the semantics of any retrieved results do. The information systems that process the results can then chose to take uncertainty explicitly into account, so that the decision-maker can weigh tradeoffs, say, risks versus costs.
3. Typically, integration over a time-line of information is required, so that a historical model of all data is important. The client should be able to integrate past, present, and simulated information, providing a continuous view, with increasing uncertainty. When delays occur in reporting past data, then the certainty at $t=0$ is already less than 1.0.
4. For true decision support multiple courses-of-action (CoAs) should be supported in the client information system, since multiple candidate alternatives may be valid simultaneously, with some probability, in the future. Implicit, for full utilization of predictive data is hence a multi-value information model. In the proverbial sense, SimQL only provides the egg here, not the chicken.
5. We do not expect to need persistent update capabilities in SimQL. Model updates are the responsibility of the providers of the simulations. The queries submitted to SimQL supply temporary variables that parameterize the simulations for a specific instance, but are not intended to update the simulation models.

While database technology provides a takeoff point for SimQL, the differences are sufficiently large that rethinking and re-engineering is warranted. We also expect feedback to occur to the traditional database domain; for instance, uncertainty is also often associated with past data, but not now treated within the database paradigm.

Specifics

The research carried out under the proof-of-concept support include three phases:

1. Wrapping several existing simulations to assess the generality of the SimQL concept
2. Defining an initial specification for SimQL and creating a simple compiler and execution support
3. Performing experiments with a variety of simulation types

This paper focuses on the language aspects, but we will first list the simulations that were wrapped in our experiments to provide information to the SimQL interface. The range illustrates that SimQL is not a point solution.

a. A spreadsheet containing formulas that projected business costs and profits into the future. Inputs were investment amounts, and results were made available for years into the future.
b. A short-range weather forecast available from NOAA on the world-wide web. Temperature and preciptation results were available for major cities, with an indication of uncertainty. The uncertainty increases rapidly beyond 5 days.
c. A long-range agricultural weather forecast for areas that overlapped with the cities. The initial uncertainty here is quite high, but increases little over a period of a several months.
d. A discrete simulation of the operation of a gasoline station, giving required refill schedules and profits.

Just as a customer application can invoke multiple databases, it can also employ multiple SimQL simulations. Our experiments only combined simulations *b*. and *c.*, selecting the forecast based on the lowest uncertainty at a selected day in the future. Still, these experiments with a few real-world simulation demonstrated the applicability of SimQL to a wide range of settings and provides a foundation for further developement of SimQL.

Language design

By borrowing ideas from SQL and the database programming paradigm we can make the schema language and the query language easy to grasp. Also, we try to preserve the simplicity of the language and its interface by providing only the minimal functionalities and data types needed. Since our experience was modest we made the language flexible and scalable for more complex simulations by taking concepts from object-oriented programming and leaving “hooks” in the language for future expansion.

While designing the language aspects, it is important to draw distinctions between the customers, builders of planning systems, wrapper developers, simulation developers, and finally SimQL system developers). The customers need only the results, and tend to be removed from direct use of SimQL and SQL. System builders, specifically builders of planning systems, will use languages as C and C++, and access simulations through SimQL and databases through SQL. People who write and maintain the actual simulations often use specialized languages. Wrapper developers write SimQL-compliant interface code for simulations, we have used C and C++ here. Wrapper generation may be allied with simulation providers who want broader audiences or with system builders who need simulation resources. Finally we represent the SimQL system developers, although we expect that in time that task will be taken on by professionals. System developers will also develop tools to aid in the creation of wrappers, as now done for non-conforming data resources [AshishK:97].

The SimQL language system shows many parallels to the database management systems. It includes catalogs, clusters, schema, and models that are analogs to SQL catalogs, clusters, schema, and views. Given this similarity, the SimQL query language employs concepts from SQL, CORBA, and KQML [FininFME:94], while the syntax of the SimQL schema language was defined using concepts from SQL, ILU, IDL, and Ontolingua [Gruber:93]. The specifics of the language as implemented can be found on our webpages, at http://www-db.stanford.edu/LIC/SimQL.html and http://www-db.stanford.edu/LIC/SimQLspec.html

Schema facilities

The task of a wrapper developer is to make a simulation schema available to builders of planning systems. After writing a SimQL wrapper for a simulation, the developer must inform the SimQL environment that such a wrapper exists. A `REGISTER` statement enables a wrapper developer to enter information about the wrapper, and hence implicitly about the wrapped simulation as metadata kept in the SimQL system. Because wrappers can come in different "shapes", we borrowed some object-oriented concepts to make the `REGISTER` statement flexible and scalable enough to handle complex wrappers. A wrapped simulation can be viewed as having a number of attributes and simulation methods. In a `REGISTER` statement, a wrapper developer can specify different `ATTRIBUTE`s of a simulator such as its performance and its accuracy in the past, and the `METHOD`s available to the customers for invoking the simulation.

Once a wrapper is registered in the SimQL metadata repository, the wrapper developer needs to create a simulation model view for each intended type of customer based on the registered wrapper. This is because

- The wrapper developer may want to expose different views in terms of attributes and methods to different customers of the same simulation [Kohavi:96].
- Various customers may have different uses for the same simulation (e.g., different inputs, different outputs, or different methods) and thus require different interfaces to the same simulation.

Therefore, a `CREATE MODEL` statement enables the wrapper developers to do just that. By using the `CREATE MODEL` statement, a wrapper developer can specify a simulation model for each customer based on the registered wrapper along with its input/output variables (specified by IN, OUT, or INOUT) and its associated method (specified in the `AS` clause). The `CREATE MODEL` statement constructs the core of the SimQL schema language. Other metadata management language statements include `DROP MODEL`, `HELP`, etc.

For instance, the model created for the business model represented in the spreadsheet only exposed the investment amounts, and the year or which the result was desired as `IN` variables, and the value of the investment, paired with its probability as the `OUT` variable. Interest rates, taxes, and business growth assumptions and their computations remained hidden from the customer, being under control of the author of the spreadsheet. Another scenario could have given the customer also a choice of investment policies.

All these SimQL schema language elements are very similar to the SQL views (i.e., `CREATE MODEL` is analogous to `CREATE VIEW`, etc.), both in syntax and concept. These similarities make the language easy to understand and expand for someone trained in database technology.

Query facilities

The next aspect was to specify the initial SimQL query language, which was built around the `SIMULATE` statement. Just as `SELECT` in SQL is used to query data in a created table or a view, `SIMULATE` in SimQL is used to invoke a simulation and obtain the results from a created simulation model. Simulation customers specify the target simulation models via the `FROM` clause, the input variables via the `WHERE` clause, and the conditions of simulation via the `HAVING` clause.

Despite all the similarities, SimQL is different from SQL in many ways, among which the following are the most prominent.

- Unlike SQL views, which are supported by real underlying SQL tables having static data, SimQL models only keeps information about interfaces to wrapped simulations.
- SimQL schema and query languages differentiate between IN, `OUT`, and `INOUT` variables which correspond to input only, output only, and input or output variables, respectively.
- Because there is uncertainty associated with any simulation, any `OUT` variable in SimQL has two parts in the form of (`value,uncertainty`), with "`value`" being the expected value of the `OUT` variable and

"`uncertainty`" the uncertainty factor for that value provided by the simulation and supplied through its wrapper.

With all the schema and query language elements sketched, we developed the language specifications for SimQL.

The SimQL environment consists essentially of a SimQL server, several SimQL clients, a interface to wrappers, several wrappers for simulations, and several actual simulators, as sketched in Figure 2. This figure combines the information flows during creation by the developer, subsequent querying by the customer, as well as the actual predictive results (bold) and possible error feedbacks (dashed).

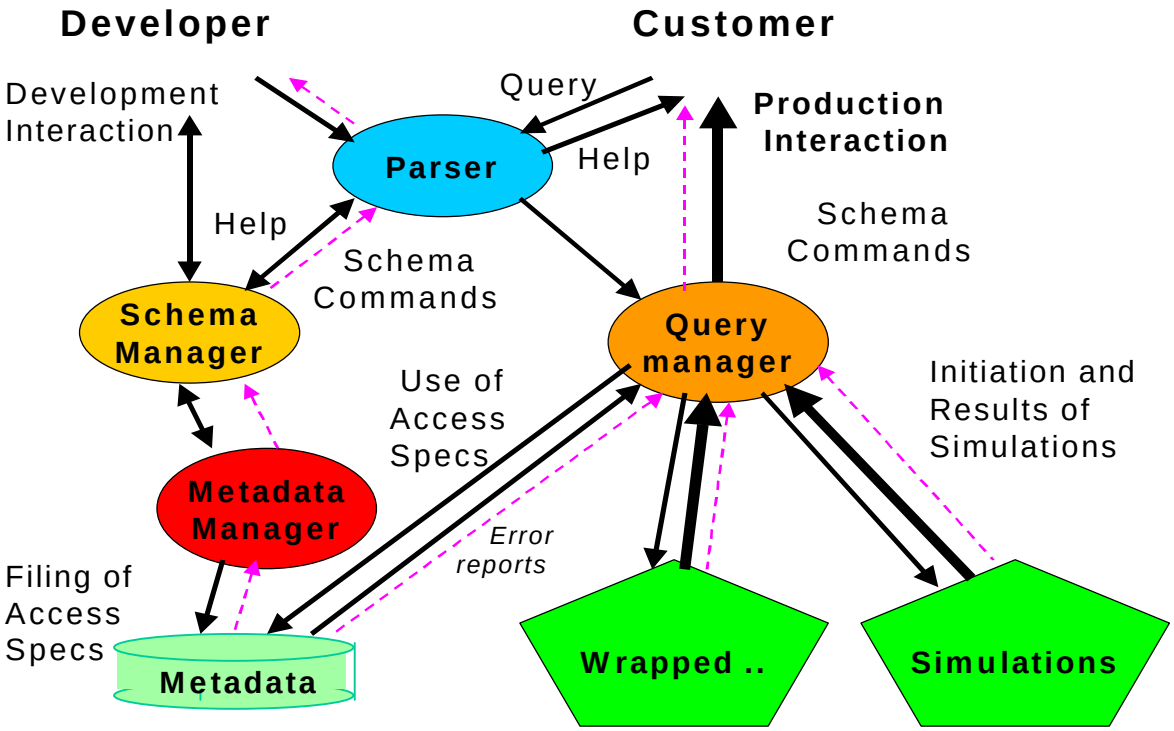

Figure 2: The SimQL prototype implementation

Implementation

The proof-of-concept implementation was achieved by modifying an existing public SQL implementation (RedBase). This approach allowed rapid implementation, although the result is not as tight as a specific implementation would have been. The benefit was to gain early on experience with compiling SimQL. The functions to be implemented were

- Parsing SimQL commands given by a customer
- Registering a wrapped simulation for a wrapper developer
- Creating simulation models for registered simulations and modifying those models arbitrarily (by the wrapper developer)
- Accessing a simulation through its model in SimQL and getting results back (by the customer)

The SimQL implementation includes four components, as depicted in Figure 2.

Written in Lex and Yacc, the SimQL parser takes SimQL statements from the customers and interprets them. After simple syntactical checking, the parser parses each statement to generate a parse tree and interprets the statement by resolving all the nodes on the tree. During the interpretation, more complex syntactical checking is performed. Depending on the type of the SimQL statement (schema vs. query), the parser packages the parsed statement accordingly and sends it to the SimQL Schema Manager or the SimQL Query Manager. A lower-level SimQL Metadata Manager was implemented to handle the file operations required by the Schema Manager and the Query Manager. The metadata files on disk store permanent information about registered wrappers and their corresponding attributes and methods, defined simulation models and their input/output variables as well as their corresponding wrappers. These metadata files are read-only to the SimQL Query Manager, which does schema lookup before accessing a required simulation.

The data structures used in all four components of the SimQL implementation originated from the SQL implementation and were adapted for simplicity. The whole implementation has about 6,000 lines of C and C++ code and is partitioned into those four modules. The SimQL Schema Manager, the SimQL Query Manager, and the SimQL Metadata Manager are written in C++, with each manager represented by a super C++ class and each SimQL statement having a method in a class. The use of object-oriented programming here has made those managers very scalable and expandable. Each of the managers can be independently compiled for testing purposes.

Results

The SimQL implementation realized the following SimQL elements/features.

- An expandable SimQL parser for parsing and interpreting SimQL commands, with robust error-checking
- A object-oriented tool for wrapper developers to `REGISTER` their wrappers
- A SimQL Schema Manager that enables the wrapper developers to use
  - `CREATE MODEL` to create simulation models
  - `DROP MODEL` to destroy created simulation models
  - A combination of `CREATE MODEL` and `DROP MODEL` to modify simulation models
- The SimQL Schema Manager allows simulation customers to use
  - `HELP` to obtain information about predefined simulation models
- A SimQL Query Manager that lets simulation customers to use `SIMULATE` to access simulation models and obtain simulation results.
- A SimQL Metadata Manager to keep track of registered wrappers and defined simulation models.

The system was tested on the wrapped weather-forecasting model in a local setting and performed as planned. To test wrapper reusability we ported the wrapper code to a second spreadsheet and determined that the adaptation to new input-output parameters was straightforward.

Assessing the current state of the world

We have focused on using simulation to assess the future. There is however an important task for SimQL in assessing the current state. Databases can never be completely current. Some may be a few minutes behind, others may be several days behind in reporting the state of resources and materiel. Intelligence information about foreign forces often lags even further behind, although it is the most crucial element in decision-making.

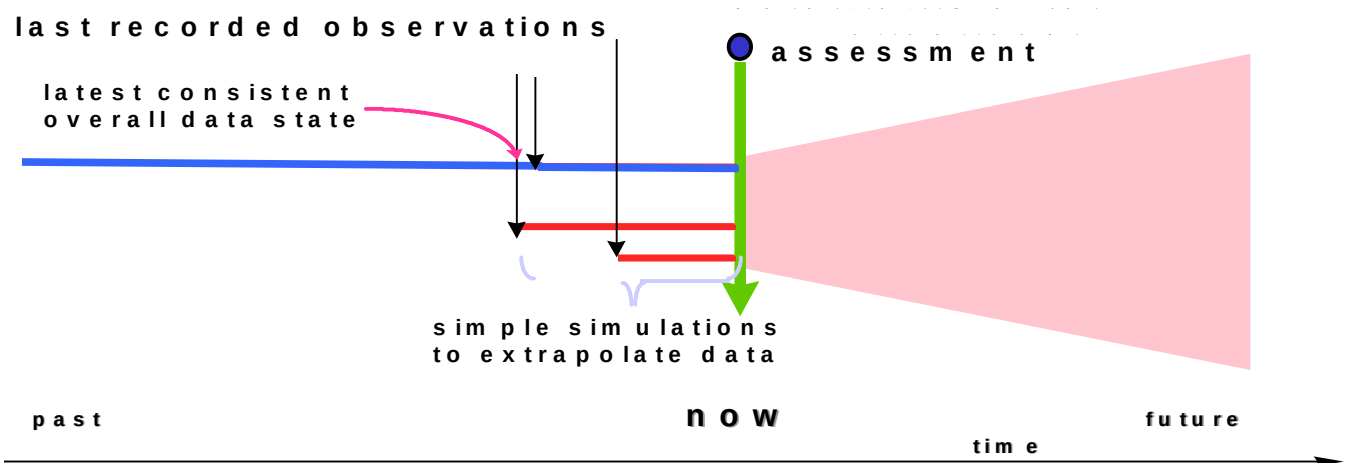

Figure 3: Even the present needs SimQL

The traditional, consistency preserving approach in database technology is to present all information at the same point in time, which reduces all information to the worst lag of all sources. It would be better to use the latest data from each source, and then project the information to the current point-in-time. In fact, that is certainly what a commander does today when faced with data of varying times of validity. SimQL can support this approach since tot provides an interface that is consistent over databases (assumed to have data with probability 1.0) and simulations, as shown in Figure 3.

These extrapolation of last know database states to the current point-in-time will help in providing a consistent, even if less certain picture of, say, where the supply trucks are now, where river crossings are stressed, and where the troops are that need the materiel. This picture is more useful than a perfect picture of the situation 2 days ago.

Future work

We have not yet transitioned SimQL to any real simulation customers and thus we do not know how receptive they will be towards the language.

- The implementation only supports basic schema and query functionalities.
- The implementation does not have an interface to a distributed simulation environment where many large-scale simulations exist.
- The implementation does not have an effective way to deal with wrappers/simulations with complex input/output data types (objects). While many real simulations have extremely complex data types (objects) that evolve in real time, the current SimQL implementation only supports the basic types used in SQL: integer, float, and text string. Object data types are desirable, but have not been well standardized. Use of an XML representation may be a solution [BeringerTJW:1998].
- Further research is needed to justify the use of some well-behaved uncertainty measure and its interaction with databases, where uncertainty often exists, but has been largely ignored [GarciaMolinaBP:92].

We plan to seek further support for the development of SimQL concepts in a setting where a realistic evaluation by potential customers can take place.

Conclusion

We have investigated the feasibility of SimQL and gained experience for a more realistic SimQL project. We have some early results, indicating that highly diverse predictive tools may be accessed in an integrated fashion via language as SimQL. Despite the limitations of our initial prototype , we believe that high-level simulation access has the potential of a major augmentation for future C4I systems. The SimQL concept is, of course, not restricted

to military simulations. An increasing number of simulations are available on the Web, but these also are difficult to integrate into information systems without an access language. Because of the importance of simulations to decision-making, we expect that concepts as demonstrated in SimQL will in time enter large-scale information systems and become a foundation that will make a crucial difference in the way that simulations will be accessed and managed. In turn, convenient access to simulations opens up new opportunities and research avenues for information systems that support decision-making.

Acknowledgments

This research was supported by DARPA DSO, Pradeep Khosla was the Program Manager; and awarded through NIST, Award 60NANB6D0038, managed by Ram Sriram. The original SQL compiler, MiniRel, was written by Mark McAuliffe, of the University of Wisconsin – Madison; and modified at Stanford by Jan Jannink and Dallan Quass under the direction of Jennifer Widom (RedBase). James Chiu, a Stanford CSD Master's student, provided and wrapped the gas station simulation. Experience in accessing the results of large, distributed simulations was gained in a related project [MalufWLP:97]. Julia Loughran of ThoughtLink provided useful comments to a presentation of this work to our sponsors [WiederholdJG:98].

References

[AshishK:97] Naveen Ashish and Craig A. Knoblock: "Semi-automatic Wrapper Generation for Internet Information Sources"; *Second IFCIS Conference on Cooperative Information Systems (CoopIS)*, Charleston, South Carolina, 1997.

[BeringerTJW:98] Dorothes Beringer, Catherine Tornabene, Pankaj Jain, and Gio Wiederhold: "A Language and System for Composing Autonomous, Heterogeneous and Distributed Megamodules"; *DEXA International Workshop on Large-Scale Software Composition*, August 98, Vienna, Austria, http://www-db.stanford.edu/CHAIMS/Doc/Papers/index.html.

[BhatnagarK:86] Bhatnagar and L.N. Kanal: "Handling Uncertain Information: A Review of Numeric and Non-numeric Methods"; in Kanal and Lemmer(eds.): *Uncertainty in AI*, North-Holland publishers, 1986.

[CliffordEa:97] James Clifford, Curtis E. Dyreson, Tomás Isakowitz, Christian S. Jensen and Richard T. Snodgrass: "On the Semantics of `Now' in Databases"; *ACM Transactions on Database Systems*, Vol. 22 No. 2, June 1997, pp. 171-214

[DateD:93] Chris J. Date and Hugh Darwen: *A Guide to the SQL Standard, 3rd ed.*; Addison Wesley, June 1993.

[FininFME:94] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire: ``KQML as an Agent Communication Language"; *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, ACM Press, November 1994.

[FishwickH:98] Paul Fishwick and David Hill, eds: *1998 International Conference on Web-Based Modeling & Simulation*; Society for Computer Simulation, Jan 1998, http://www.cis.ufl.edu/~fishwick/webconf.html.

[FullerMP:93] David A. Fuller, Sergio T. Mujica, José A. Pino: "The Design of an Object-Oriented Collaborative Spreadsheet with Version Control and History Management"; SAC'93, Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing: States of the art and practice, pp. 416-423.

[GarciaMolinaBP:92] Hector GarciaMolina, D. Barbara, and D. Porter: "The Management of Probabilistic Data"; *IEEE Transactions on Knowledge and Data Engineering*, Vol.4, No. 5, October 1992, pp. 487-502.

[Gruber:93] Thomas R.Gruber: ``A Translation Approach to Portable Ontology Specifications"; *Knowledge Acquisition*, Vol.5 No. 2, pp.199--220, 1993

[HammerEa:97] J. Hammer, M. Breunig, H. Garcia-Molina, S. Nestorov, V. Vassalos, R. Yerneni: "Template-Based Wrappers in the TSIMMIS System"; *ACM Sigmod 26*, May 1997.

[IEEE:98] *P1561, Draft IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)*; IEEE, 1998.

[INEL:93] Idaho National Engineering Laboratory: "Ada Electronic Combat Modeling"; *OOPSLA'93 Proceedings*, ACM 1993.

[Jiang:96] Rushan Jiang: Report on the SimQL project; submitted to Prof. Wiederhold, CSD Stanford, August

[LindenG:92] Ted Linden and D. Gaw 1992: "JIGSAW: Preference-directed, Co-operative Scheduling," *AAAI Spring Symposium: Practical Approaches to Scheduling and Planning*, March 1992

[Kohavi:96] Ron Kohavi: Wrappers for Performance Enhancement and Oblivious Decision Graphs; PhD thesis, Stanford University CSD, 1996.

[MalufWLP:97] David A. Maluf, Gio Wiederhold, Ted Linden, and Priya Panchapagesan: "Mediation to Implement Feedback in Training"; *CrossTalk: Journal of Defense Software Engineering*, Software Technology Support Center, Department of Defense, August 1997.

[McCall:96] Gene McCall (editor): *New World Vistas, Air and Space Power for the 21st Century*; Air Force Scientific Advisory Board, April 1996, Information Technology volume, pp. 9.

[Miller:56] George Miller: "The Magical Number Seven ± Two"; *Psych.Review*, Vol.68, 1956, pp.81-97.

[MillerT:95] Duncan C. Miller and Jack A. Thorpe: "SIMNET: The Advent of Computer Networking"; *Proceedings of the IEEE*, August 1995, Vol.83 No.8, pages 1116-1123.

[Orsborn:94] Kjell Orsborn: "Applying Next Generation Object-Oriented DBMS for Finite Element Analysis"; ADB conference, Vadstena, Sweden, in Litwin, Risch: *Applications of Database'*, Lecture Notes In Computer Science, Springer, 1994.

[Singhal:96] Sandeep Singhal: *Effective Remote Modeling in Large-Scale Distributed Interactive Simulation Environments*; PhD Thesis, Stanford CSD, 1996.

[Snodgrass:95] Richard T. Snodgrass (editor): *The TSQL2 Temporal Query Language*; Kluwer Academic Publishers, 1995,

[Wiederhold:93] Gio Wiederhold: "Intelligent Integration in Simulation"; MORS Mini-symposium, Fairfax VA, Military Operations Research Society, Alexandria VA, November 1993.

[WiederholdG:97] Gio Wiederhold and Michael Genesereth: "The Conceptual Basis for Mediation Services"; *IEEE Expert, Intelligent Systems and their Applications*, Vol.12 No.5, Sep-Oct.1997.

[WiederholdJG:98] Gio Wiederhold, Rushan Jiang, and Hector Garcia-Molina: “An Interface for Projecting CoAs in Support of C2; *Proc.1998 Command & Control Research & Technology Symposium*, Naval Postgraduate School, Monterey CA, June 1998, pp.549-558.

[Widom:97] Jennifer Widom: "Research Problems in Data Warehousing"; *Proceedings of the 4th Int'l Conference on Information and Knowledge Management (CIKM)*, November 1995.