

SimQL: Simulation-Based Decision Modeling Over Stochastic Databases

Susan FARLEY^{a,1}, Alexander BRODSKY^a, John MCDOWALL^a and Nathan EGGE^a

^a*Department of Computer Science
George Mason University, Fairfax, Virginia*

Abstract. In this paper we propose an extension of SQL with probability functions expressed through stochastic simulation, its canonical implementation and an efficient algorithm for top-k decisions.

Keywords. Decision-guidance management systems, stochastic simulation, uncertain data, optimal computation budget allocation

Introduction

Making stochastic decisions is a part of everyday life. For example, suppose that Camford University, located on the coast must always have a consistent energy supply. However, the local power grid cannot supply the necessary uninterrupted power due to transformer burnout, wild fires or other natural disasters. Camford wants to establish backup sources of energy using wind for when the public electrical grid fails. The potential savings will be uncertain due to the variability of wind and the uncertainty of sufficient power generation.

Even with a considerable amount of data to help with the decision making, there are still problems to overcome. Much of this data can have a probabilities attached to it, be statistical in nature, or can be stochastic. Relational databases do not process probabilistic data and do not have the additional functionality needed for analyzing statistical data or defining stochastic attributes. To overcome these limitations, considerable research has been done in the field of probabilistic and statistical databases as well as stochastic simulation, but this work also has limitations.

Probabilistic databases are databases capable of managing large amounts of uncertain data [3]. Each tuple has a probability associated with it corresponding to its likelihood. All possible answers (worlds) have to be modeled in the database, but computing the exact confidence of the resulting tuples for a query is an NP complete problem. As a result, researchers have to use approximation techniques [4].

Statistical databases are usually relational databases containing statistical data that have been extended to allow for advanced statistical analysis techniques. Unfortunately, the quality of data is not always optimal and there are security problems resulting in either the queries being too restrictive or the users possibly getting private information that they should not have access to [5].

The shortcoming with probabilistic and statistical databases is that they deal with explicit probability distributions, whereas in many applications probabilistic

¹ Corresponding Author.

distributions are complex and implicitly defined. For instance the amount of power that can be wind generated for Camford as well as the possible savings from using wind generated power versus the more conventional means of backup can be difficult to define with explicit probability distributions. Stochastic simulation allows a user to create a complex model to simulate an event and serves as an implicitly defined stochastic model. However, it is time consuming to make decisions based on trial and error, even after applying heuristics to speed up the process.

We propose to bridge this gap and allow the user to make decisions using queries similar to what they are familiar with using in standard relational databases, yet with stochastic attributes implicitly defined as a simulation.

The goal of this research is to investigate whether the SQL language can be extended to incorporate stochastic attributes in relational databases. The contributions of this paper are the descriptions of the syntax and semantics of SimQL along with the baseline implementation of SimQL and a brief explanation of the Optimal Computing Budget Allocation (OCBA). In Section 1, we explain the syntax, semantics, and canonical implementation of SimQL. In Section 2 we explain the OCBA as used for Top-k queries and then we conclude the paper in Section 3.

1. SimQL: Syntax, Semantics, and Canonical Implementation

Though the initial problem of whether there is enough consistent wind available and how many turbines would be needed to support the needs of Camford could be solved with a simple MATLAB simulation, Camford also wants to be able to justify the cost of installing the wind turbines versus another backup strategy such as diesel generators. The university also needs to ensure that there continues to be enough power in their grid to support the growth of the university and to adjust for changes in the national grid without having to recreate the model each time a variable changes or a new question needs to be addressed without making decisions based on trial and error. Since most of the data needed (i.e. average daily wind speed, daily base electric usage, daily peak electric usage) can easily be obtained and stored in a database, SimQL makes answering these questions at hand along with the expected questions in the future easy.

For instance, a view with a stochastic attribute of the output of electricity from the wind based on the average annual wind speed, along with other attributes like the wind turbine hub height, a deterministic attribute in the table `wind_power_info`, can be defined as:

```
CREATE OR REPLACE VIEW simql.vwWndPwr AS
  SELECT wind_power_id, location,
         annual_avg_speed,
         simpleSim (annual_avg_speed) AS wndout
  FROM simql.wind_power_info;
```

This view uses the user defined function `simpleSim`. This function was written as part of a Java library that was imported for use in the database. When materialized, this view will return a value based on the input into this function.

Once a view has been created, the user can query the view like a standard table, materializing the stochastic attribute into a deterministic attribute (possible world).

```
SELECT location, annual_avg_speed,
       SimQL.exp( 'wndout', 'vwWndPwr' ,
```

```

                                wind_power_id)
FROM SimQL.vwWndPwr;

```

The query above will return the location, annual average wind speed, and the approximation of the expected wind output after the simulation has been run for the allocated budget. A query that will return the variance of the output after the simulation has been run until the desired stopping point has been reached would look something like the following:

```

SELECT location, annual_avg_speed,
       simql.variance('wndout', 'vwWndPwr',
                    wind_power_id)
FROM simql.vwWndPwr;

```

To return the probability that the tuple is going to be in the top ten locations based on the output, the following query would be used:

```

SELECT location, annual_avg_speed,
       simql.topk('wndout', 'vwWndPwr',
                wind_power_id, 10)
FROM simql.vwWndPwr;

```

These functions can also be used in the WHERE clause to eliminate tuples that do not fit the desired criterion. The following query would only return the tuples where the probability that the expected output is greater than 8600 is more than eighty percent.

```

SELECT location, annual_avg_speed
FROM simql.vwWndPwr
WHERE simql.prob('wndout' > 8600, 'vwWndPwr',
                wind_power_id) > .8;

```

A prototype of the language SimQL has been integrated with the open source database PostgreSQL using the relational data model with the addition of stochastic attributes as described in [1]. Basically, SimQL is an extension of standard SQL language that allows a user to run a simulation based on known attributes stored in a standard data table to compute the value of an unknown attribute through stochastic simulation. For this purpose, four new functions are defined: EXP, VARIANCE, PROB and TOPK. Each runs the simulation for the stochastic variable using the parameters set in the sim_config table with EXP returning the *approximation of expectation* of the stochastic attribute, VARIANCE returning the *variance* of the stochastic attribute, PROB returning the *probability* that the stochastic attribute satisfies the inequality equation, and TOPK returning the *probability* that the tuple is in the Top-k for the stochastic attribute.

EXP, VARIANCE, and PROB use a canonical evaluation based on Monte Carlo and TOPK uses Optimal Computing Budget Allocation (OCBA).

2. Top-k Algorithm Based on Simulation Budget Optimization

Simulations are popular for finding solutions to large, complex stochastic problems such as the ones facing Camford. The difficulty is the computational power required to run these simulations with the different possible solutions that exist. The Optimal Computing Budget Allocation (OCBA) algorithm [2] allows for the optimal allocation of a computational budget when selecting the Top-k tuples in a query as well as the ability to attain a certain probability that a row is contained in the Top-k with minimal simulations. After an initialization run, a portion of the computational budget is

assigned to each tuple for additional simulation runs. Using OCBA ensures that a larger portion of the given computational budget is allocated to the tuples that have a higher variance and that tuples that are further away from the Top-k “cut-off” have a smaller budget. This ensures that tuples with a higher prospect of “jumping the line” so that they would be included in the Top-k results as well as tuples that could be downgraded into not being included in the Top-k get more attention than tuples that do not have any chance of switching from their current designation. After each portion of the computational budget has been utilized, OCBA is used again to allocate more of the computational budget if the desired degree of certainty that the selected rows are indeed in the Top-k has not been reached. This process continues until either the computational budget has been depleted or the desired probability has been reached.

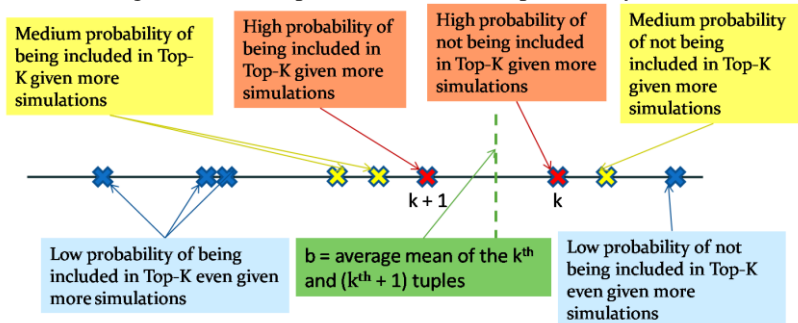


Figure 1 OCBA Technique.

3. Conclusions and Future Work

SimQL allows the user to define computationally complex simulations that can be run for stochastic attributes while allowing the user to alter the parameters. SimQL will leverage the power of the relational database to investigate multiple possible decisions in rapid succession without the need to create or update a complex model using Matlab or other traditional stochastic modeling techniques. SimSQL can easily be applied to many areas where decision making is based on traditional simulation techniques, such as emergency responder scenarios and manufacturing. We hope that future iterations of SimQL will include budget optimizations for special cases of problems where the expression structure allows the answer to be precisely computed analytically so that simulation would not be required.

References

- [1] Alexander Brodsky, X. Sean Wang, "Decision-Guidance Management Systems (DGMS): Seamless Integration of Data Acquisition, Learning, Prediction and Optimization," hiecss, pp.71, Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008), 2008.
- [2] Chen, C. H., He, D., Fu, M. C., and Lee, L. H. 2008. Efficient simulation budget allocation for selecting an optimal subset. *INFORMS J. Comput.* 20, 579-595.
- [3] V. Biazzo, A. Ferro, A. Gilio, & R. Guigno, A general probabilistic database model (2009).
- [4] C. Koch & D. Olteanu, Conditioning Probabilistic Databases. *Computing*, 313-325.
- [5] J. Gehrke, Data Mining for Security and Privacy, *Security(2002)*, 14853-14853.
- [6] L. Wong & G. C. Kooten, Economic Aspects of Wind Power Generation in Developing Countries, *Comparative and General Pharmacology*, (September 2009).