# SQL for SRL: Structure Learning Inside a Database System
# Position Paper

Oliver Schulte and Zhensong Qian
oschulte,zqian@sfu.ca

School of Computing Science
Simon Fraser University
Vancouver-Burnaby, Canada

## Abstract

The position we advocate in this paper is that relational algebra can provide a unified language for both representing and computing with statistical-relational objects, much as linear algebra does for traditional single-table machine learning. Relational algebra is implemented in the Structured Query Language (SQL), which is the basis of relational database management systems. To support our position, we have developed the FACTORBASE system, which uses SQL as a high-level scripting language for statistical-relational learning of a graphical model structure. The design philosophy of FACTORBASE is to manage statistical models as first-class citizens inside a database. Our implementation shows how our SQL constructs in FACTORBASE facilitate fast, modular, and reliable program development. Empirical evidence from six benchmark databases indicates that leveraging database system capabilities achieves scalable model structure learning.

## Introduction

The statistical analysis of structured data requires structured machine learning models. Database researchers have developed a system architecture where statistical models are stored as first-class citizens inside a relational database management system (RDBMS) (Wang et al. 2008; Niu et al. 2011). The goal is to seamlessly integrate query processing and statistical-relational inference *inside the database*, rather than invoking external procedures. These systems focus on inference *given* a statistical-relational model, not on *learning* the model from the data stored in the RDBMS. We describe the FACTORBASE system that complements the in-database probabilistic inference systems with an in-database probabilistic model learning system. The name FACTORBASE indicates that our system supports learning a set of (par)-factors for a log-linear multi-relational model, typically represented in a graphical model (Kimmig, Mihalkova, and Getoor 2015).

There are several previous systems that leverage RDBMS support for learning (Hellerstein et al. 2012; Kraska et al. 2013; Deshpande and Madden 2006), but they apply to traditional learning where the data are represented in a *single* table or data matrix. The novel contribution of FACTORBASE is supporting graphical model learning for *multi-relational* data stored in different interrelated tables. The Sindbad system (Wicker, Richter, and Kramer 2010) provides support for some multi-relational knowledge discovery tasks in an inductive database, but not for graphical model construction. Multi-relational graphical model construction raises several special challenges, such as:

1. A description language for specifying metadata about structured random variables.
2. Efficient mechanisms for constructing, storing, and transforming complex statistical objects, such as cross-table sufficient statistics, parameter estimates, and model selection scores.
3. Computing model prediction scores for relational test instances.

FACTORBASE applies SQL as a scripting language to implement database services that provide these capabilities. While FACTORBASE provides a good solution for each of these system capabilities in isolation, the ease with which large complex statistical-relational objects can be integrated via SQL queries is a key feature.

### Evaluation

Our system is fully implemented and source code is available available on-line (Qian and Schulte 2015). We summarize the evaluation of FACTORBASE on six benchmark databases. For each benchmark database, the system applies the learn-and-join algorithm, a state-of-the-art SRL algorithm that constructs a statistical-relational Bayesian network model (Schulte and Khosravi 2012). The learned Bayes net structure can be converted to a Markov Logic network structure or a set of clauses (Khosravi et al. 2010). The same SQL scripts work for all benchmark databases, which demonstrates the generality of our approach.

Our experiments show that FACTORBASE pushes the scalability boundary: Learning scales to databases with over $10^6$ records, compared to less than $10^5$ for previous systems. At the same time it is able to discover more complex cross-table correlations than previous SRL systems. The scalability improvement is mainly due to the efficient computation and caching of sufficient statistics supported by SQL. Our experiments focus on two key services for an SRL client: (1) Computing and caching sufficient statistics, (2) computing model predictions on test instances. The system can handle

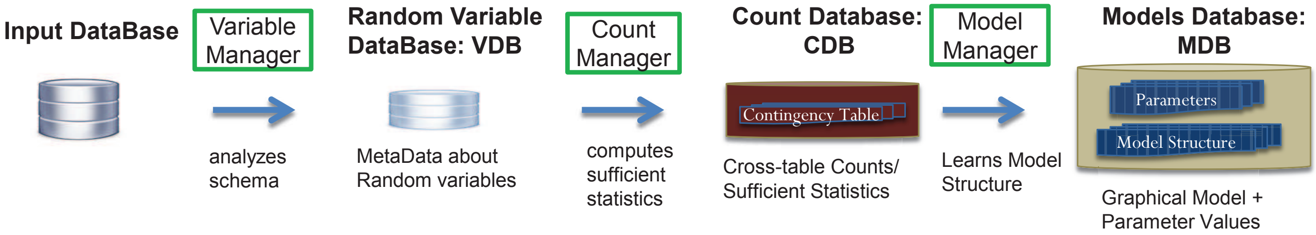

Figure 1: System Flow. All statistical objects are stored as first-class citizens in a DBMS. Objects on the left of an arrow are utilized for constructing objects on the right. Statistical objects are constructed and managed by different modules (boxes).

as many as 15M sufficient statistics. SQL facilitates block-prediction for a set of test instances, which leads to a 10 to 100-fold speedup compared to a simple loop over test instances.

## Benefits

We advocate using SQL as a high-level scripting language for SRL, because of the following advantages.

1. Extensibility and modularity, which support rapid prototyping. SRL algorithm development can focus on statistical issues and rely on the RDBMS for data access and processing.
2. Increased scalability, in terms of both the size and the complexity of the statistical objects that can be handled.
3. Generality and portability: standardized database operations support "out-of-the-box" learning with a minimal need for user configuration.

# System Overview

Our system design represents statistical objects as relational tables, on a par with the original data tables, so that SQL can be used to manage them. Figure 1 represents key system components. The starting point is a multi-relational database containing the input data.

## System Components

**The Schema Analyzer** The schema analyzer is an SQL script that queries the system catalog table to define a default set of relational random variables (par-RVs) for statistical analysis (Kimmig, Mihalkova, and Getoor 2015). The metadata include the domain of the par-RVs (possible values), and type information (possible arguments). The schema analyzer extracts metadata about the random variables from the database system catalog. The random variables and associated metadata are stored in the **random variable database** *VDB*. We highlight two features of the *VDB* component.

(i) The set of par-RVs and the associated metadata is constructed *automatically* from the input database. Thus FACTORBASE utilizes the data description resources of SQL to faciliate the "setup task" for relational learning (Walker et al. 2010).

(ii) Representing metadata explicitly offers two advantages. First, a user can easily edit the *VDB* to customize the learning behavior, for instance by deleting irrelevant par-RVs. Second, it is possible to export metadata from other formats to the *VDB* format. In this way FACTORBASE can serve as a structure learning backend to expressive specification languages for other relational models (Guazzelli et al. 2009; Milch et al. 2005).

**The Count Manager** A key service for statistical-relational learning is counting how many times a given relational pattern (par-RV) is instantiated in the data. Such counts are known as *sufficient statistics*. Accessing sufficient statistics is often the main scalability bottleneck. The access patterns of a model search procedure are inherently sequential and random (Niu et al. 2011), and therefore it is important to cache sufficient statistics. Caching is even more important if the data is stored on disk in an RDBMS, rather than in main-memory. There are several reasons for employing an RDBMS for gathering sufficient statistics. (1) The machine learning application saves expensive data transfer by executing count operations in the database server space rather than local main memory. (2) SQL optimizations for aggregate functions such as SUM and COUNT can be leveraged. (3) Sufficient statistics can be stored in the RDBMS. For many datasets, the number of sufficient statistics runs in the millions and is too big for main memory. A novel aspect of FACTORBASE is managing *multi-relational sufficient statistics* that combine information *across* different tables in the relational database. This requires combining SQL aggregate functions with table joins (Qian, Schulte, and Sun 2014).

**The Model Manager** The Model Manager supports the construction and querying of large structured statistical models, which are stored in the **Model Database** MDB. Services provided by the Model Manager include the following. (1) Compute parameter estimates for the model using the sufficient statistics in the Count Database. (2) Computing model characteristics such as the number of parameters or degrees of freedom in a model. (3) Computing a model selection score that quantifies how well the model fits the multi-relational data. Model selection scores are usually functions of the number of parameters and the parameter estimates. By employing the SQL view mechanism, parameter estimates and model selection scores are updated automatically during the model search.

# References

Deshpande, A., and Madden, S. 2006. MauveDB: supporting model-based user views in database systems. In *SIGMOD*, 73–84. ACM.

Guazzelli, A.; Zeller, M.; Lin, W.-C.; and Williams, G. 2009. Pmml: An open standard for sharing models. *The R Journal* 1(1):60–65.

Hellerstein, J. M.; Ré, C.; Schoppmann, F.; Wang, D. Z.; Fratkin, E.; Gorajek, A.; Ng, K. S.; Welton, C.; Feng, X.; Li, K.; and Kumar, A. 2012. The MADlib analytics library: Or MAD skills, the SQL. *PVLDB* 5(12):1700–1711.

Khosravi, H.; Schulte, O.; Man, T.; Xu, X.; and Bina, B. 2010. Structure learning for Markov logic networks with many descriptive attributes. In *AAAI*, 487–493.

Kimmig, A.; Mihalkova, L.; and Getoor, L. 2015. Lifted graphical models: a survey. *Machine Learning* 99(1):1–45.

Kraska, T.; Talwalkar, A.; Duchi, J. C.; Griffith, R.; Franklin, M. J.; and Jordan, M. I. 2013. MLbase: A distributed machine-learning system. In *CIDR*.

Milch, B.; Marthi, B.; Russell, S. J.; Sontag, D.; Ong, D. L.; and Kolobov, A. 2005. BLOG: probabilistic models with unknown objects. In *IJCAI-05*, 1352–1359.

Niu, F.; Ré, C.; Doan, A.; and Shavlik, J. W. 2011. Tuffy: Scaling up statistical inference in Markov Logic Networks using an RDBMS. *PVLDB* 4(6):373–384.

Qian, Z., and Schulte, O. 2015. The BayesBase system. `www.cs.sfu.ca/~oschulte/BayesBase/BayesBase.html`.

Qian, Z.; Schulte, O.; and Sun, Y. 2014. Computing multi-relational sufficient statistics for large databases. In *CIKM*, 1249–1258. ACM.

Schulte, O., and Khosravi, H. 2012. Learning graphical models for relational data via lattice search. *Machine Learning* 88(3):331–368.

Walker, T.; O'Reilly, C.; Kunapuli, G.; Natarajan, S.; Maclin, R.; Page, D.; and Shavlik, J. W. 2010. Automating the ILP setup task: Converting user advice about specific examples into general background knowledge. In *ILP*, 253–268.

Wang, D. Z.; Michelakis, E.; Garofalakis, M.; and Hellerstein, J. M. 2008. BayesStore: managing large, uncertain data repositories with probabilistic graphical models. In *VLDB*, volume 1, 340–351.

Wicker, J.; Richter, L.; and Kramer, S. 2010. SINDBAD and SiQL: Overview, applications and future developments. In Dvzeroski, S.; Goethals, B.; and Panov, P., eds., *Inductive Databases and Constraint-Based Data Mining*. Springer New York. 289–309.