Capturing the Laws of (Data) Nature

Hannes Mühleisen
CWI, Amsterdam
hannes@cwi.nl

Martin Kersten
CWI, Amsterdam
mk@cwi.nl

Stefan Manegold
CWI, Amsterdam
manegold@cwi.nl

ABSTRACT

Model fitting is at the core of many scientific and industrial applications. These models encode a wealth of domain knowledge, something a database decidedly lacks. Except for simple cases, databases could not hope to achieve a deeper understanding of the hidden relationships in the data yet. We propose to harvest the statistical models that users fit to the stored data as part of their analysis and use them to advance physical data storage and approximate query answering to unprecedented levels of performance. We motivate our approach with an astronomical use case and discuss its potential.

1. MOTIVATION

> "Essentially, all models are wrong, but some are useful." – George E. P. Box

The realities of analytical data management have shown that the various bottlenecks in hardware architecture often forbid reading all formally relevant data, especially if a result is to be available in reasonable time. But exact answers are not always required, in particular for interactive data exploration applications. Two approaches for approximate query answering come to mind, sampling and synopses. In sampling, only a subset of data is used to answer a time-critical query [16, 2]. Doing so will introduce errors in the result, but predicting the extent of these errors is well understood. Synopses are compressed lossy approximations of the data, much like a JPEG approximation of an exact bitmap. Approximate queries can use these synopses instead of the actual data [8, 1]. At the same time, the query workload is used to make efficiency-relevant storage decisions such as whether to store data in row-major or column-major representations [4]. These methods suffer from a common flaw: They cannot hope to achieve a higher-level understanding of the data, and we need to go further than generic lossy compression, mining of functional dependencies, or query log harvesting.

7th Biennial Conference on Innovative Data Systems Research (CIDR '15) January 4-7, 2015, Asilomar, California, USA.

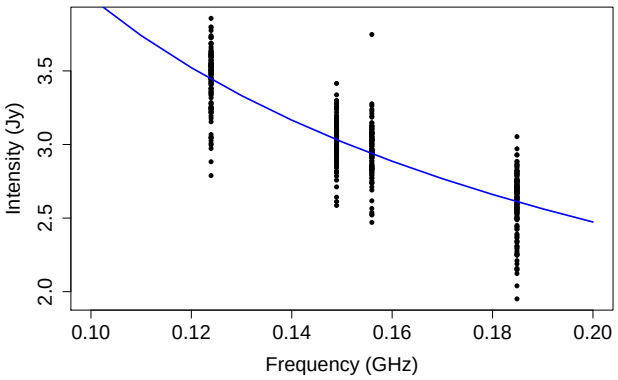

Figure 1: Raw data vs. Model: LOFAR

In our opinion, we have been overlooking one of the most common uses of raw data, in particular for scientific and statistical data management: The comparison of collected observations with statistical models ("fitting").

It is this comparison between nature and our understanding of it that forms the core of natural sciences [21, 18]. If the data fits the model, and the model is even able to predict future observations, we can assume to have understood the phenomenon in question [13, 12]. Once a model is found, observations where nature significantly deviates from predictions often become the focus of interest. For example, the search for exoplanets relies on analyzing a minute amount of wobbling in a star's position or a tiny dip in its intensity.

Currently, data management and model fitting are decoupled, the fitting process reads the data once and later discards it. Data management is unaware of the user-supplied model that is compared with the stored observations. This is unfortunate, as we believe that these models are invaluable for data management itself. In particular, they may hold the key for unprecedented performance in approximate query answering. User models can provide approximations in a similar way to the data synopses discussed before, but with higher accuracy. The user model encodes much of the domain knowledge and experience required to understand the data at hand. Also, a deeper insight into the relationships within the stored data allows more informed decisions regarding storage operations such as compression.

Source	Wavelength ν	Intensity I		Source	Spectral Index α	Constant p	Residual SE
1	0.1559555	0.2315911		1	-0.7183309	0.06257838	0.006559710
1	0.1239243	0.3478159	$\Rightarrow$	2	-0.8932245	0.07195620	0.008007786
1	0.1489243	0.1592717		3	-0.7880774	0.56190180	0.016778232
[1,452,821 more rows]				[35,681 more rows]			

Table 1: Example LOFAR observations and approximation

Our vision is a database system which is able to gain unprecedented understanding by autonomous and proactive harvesting of statistical models as they are fitted to the stored data. Rather than forcing users to provide detailed specification of the relationships within the data, we provide means for model fitting inside the database. This gives the system access to the statistical model provided by the user, which can be transparently stored, re-executed, and generally employed for approximate query answering and data storage optimization.

In the remainder of this paper, we motivate our approach with an example and discuss the general process of model fitting and their capture. Then, we present general opportunities and challenges for a database system which harvests user models autonomously. Finally, we conclude with related and future work.

2. REAL-WORLD EXAMPLE: LOFAR TRANSIENTS

The Low-Frequency Array (LOFAR) is a large-scale telescope for radio astronomy. It consists of 48 antenna stations distributed across several countries in Europe [20]. By combining signals from all stations, the telescope generates radioastronomical "images" of the sky. The LOFAR Transients Key Science project tries to find the astronomical phenomena ("sources") that show fluctuations in their signals, for example pulsars, quasars, black holes and afterglows of gamma-ray bursts. To achieve this, observations from sources are correlated over multiple images and their intensities recorded. In general, all observations are subject to a large amount of interference.

We have obtained a small sample of 1,452,824 measurements from 35,692 sources to serve as example data set. The data set contains a source identifier, the frequency of the observation and the observed source intensity ("flux"). Table 1 shows some example values. To a database, this is a three-column relational table, where the source identifier is an integer and the others are floating-point numbers. From a model perspective, the source identifier and the frequency are input variables, and the flux is the observed output. Furthermore, our understanding of astronomical radio sources predicts that within radio frequencies, the intensity I of any individual source is proportional to on the observation frequency ν and a source-specific "spectral index" α: $I \propto \nu^{\alpha}$ [6].

Now imagine that an astronomer fits this model to the observational data. Even though we assume most sources follow the predictions, their parameters will still vary widely. The result of the fitting process is the α constant for each source and another constant p to express the proportionality. We thus obtain another table that contains these constants for each of the 35,692 sources. In terms of bytes, we were able to replace ca. 11MB of observations with 640KB of model parameters, ca. 5% of the original dataset size.

Figure 1 shows the result of the model fitting for a single LOFAR source: We can see the widely varying observations for the four different observed frequency bands in the dataset. The blue line indicates the result of the model fitting process. We predict a spectral index of -0.69 for this source, which indicates that these observations are the result of thermal emissions (e.g. from a star). There are, of course, other sources that are known to not adhere to the model, e.g., have turn-overs in their spectral index. However, these are not only rare, but can now be spotted much easier by observing the goodness-of-fit for the model.

As can be seen in Table 1, the model parameter table also contains a measure of the goodness of fit of the model to the data as the residual standard error. This measure serves two purposes. One, it allows us to annotate data approximated through the model with an indication of the error that is to be expected. Second, the user will be interested in these values to determine whether the chosen model fits the data well. In this particular application, it is unlikely that the number of sources increases without changes to the telescope. It is however to be expected that the amount of measurements will grow linearly over time. Therefore, in this approximation, if ten times more observations per source are collected, the model will only get more precise, not larger in terms of storage or processing requirements.

Given the model and the parameters table, we can now reconstruct tuples and thus approximately answer queries on the original table. For example, consider the two following SQL queries:

```
SELECT intensity FROM measurements
 WHERE source = 42
 AND wavelength = 0.14;

SELECT source, intensity FROM measurements
 WHERE wavelength = 0.14
 AND intensity > 3.0;
```

The first query requires us to look up the two parameters to the model function $I = p * \nu^{\alpha}$ and evaluate the function with those parameters. Then, the result can be calculated. A method to approximately answer the second query would be to calculate the intensity as an intermediate step by calculating all intensity values with the stored set of parameters for all sources and the given wavelength. Regardless, we can approximately answer both query answers solely from the model data. For some classes of models, an analytical solution would also be feasible. We will discuss these challenges in the following section.

3. MODEL FITTING AND CAPTURE

As could be seen from our example, models encode a hypothesis about the relationship of various parameters. As such, they consist of two parts, an arbitrary function of the input variables and various constant but unknown parameters. The task of the fitting process is to approximate those parameters. This is performed using a set of observed (sampled) input and output values. We can calculate a set of predicted output values, and compare them to the observed output values. In general, we need more observed input/output pairs than model parameters to find a set of fitted parameters. We do not make any restrictions on the type of model to be fitted, but there are two major classes from the viewpoint of the algorithm used to fit the parameters to the observations:

In the simpler case of linear models ($\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$), we can use the ordinary least squares method to find an *analytical* solution for the unknown parameters $\boldsymbol{\beta}$ by minimizing the sum of squared residuals. This can be done by solving the linear equation system $\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$. Since solving linear equations is a well-understood algorithmic problem, we can calculate the model parameters in this case [21].

Contrary, in the general (non-linear) case, we have to fall back to optimization algorithms. For example, the Gauss-Newton algorithm is an iterative method to find the set of parameters for which the predicted output values that have the least squared distance from the observed output values. Formally the parameters β are approximated as follows: $\boldsymbol{\beta}^{(s+1)} = \boldsymbol{\beta}^{(s)} - \left(\mathbf{J_r}^{\top}\mathbf{J_r}\right)^{-1}\mathbf{J_r}^{\top}\mathbf{r}(\boldsymbol{\beta}^{(s)})$ with $\mathbf{J_r} = \frac{\partial r_i(\boldsymbol{\beta}^{(s)})}{\partial \beta_j}$ and r being a set of arbitrary functions of the parameters β. It is far from certain that this or related algorithms will find the perfect set of parameters, and their convergence can be highly dependent on the choice of starting parameters. Also, it is possible that the optimization algorithm gets trapped in local extrema.

However, since it is the user who supplies the models, we can ignore the computational complexity and convergence issues of the fitting process here. It is their responsibility to design a model, choose the appropriate optimization method, and choose a set of starting parameters that will lead to convergence.

A practical issue concerns the expression of the models and the fitting process itself: Statistical environments have facilities to let users express and fit arbitrary models by programming, but not relational databases. But in order to achieve our goal of interleaving model fitting and querying, it would be practical to use the same representation. Fortunately, a recent trend moved statistical processing closer to the data. Major commercial databases now embed the R statistical environment within the database itself [11, 14]. This also helps to solve a major performance issue: Transferring all data from the database to the statistical environment is not necessary any more. At the same time, we can store the models in their source code form inside the database, since the execution environment for the code is also available.

Model fittings takes place in statistical environments and not in data management systems. It is unrealistic to assume this to change. However, in previous work, we have demonstrated

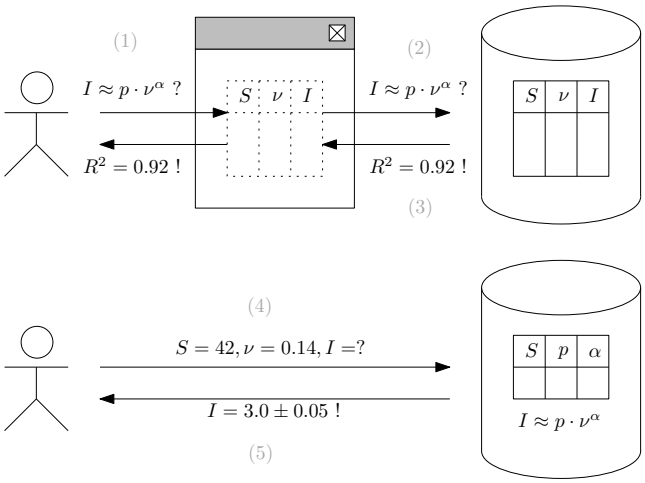

Figure 2: Model Interception

how complex statistical calculations in R can be transparently shipped off to analytical databases without changing the user experience [10]. This is achieved by constructing a so-called "strawman object" in the statistical environment, which wraps a database table or query result, but is indistinguishable from a local dataset. Any command the user performs on this object is forwarded to the data management system. We propose to leverage this method for model fitting as well. Together with the aforementioned integration of statistical environments into data management systems, this can create a win-win situation, where the user benefits from faster analysis, and the data management system benefits from the user models.

Figure 2 illustrates this approach: A user wishes to fit a model against a dataset in a statistical environment (1). The data there is however a strawman for a database table. Therefore, the fitting process gets offloaded there (2). The database dutifully fits the model and returns the goodness of fit (3). At the same time, the database stores the model as well as its parameters for later use. In the next interaction, the user queries the database for a value that can be approximately reconstructed using the stored models (4). This value is calculated using the model and the small parameter dataset and returned with error bounds (5).

Since the entire process runs inside the database, we can intercept fitting, determine the accessed data, and judge the quality of the fitted model. For example, we could use the R^2 coefficient of determination or the results of an F-test against a model with fewer parameters [21].

Overall, we therefore assume that we have the means to

1. Fit a user-supplied model to data stored in a relational database.
2. Judge the quality of the model.
3. Store the model itself and the trained parameters.
4. Evaluate the model on different input values using the trained parameters.

	Physical Storage	Approximate Queries
Opportunities	"True" semantic compression Zero-IO scans	Analytic solutions for linear models Model exploration Data anomalies
Challenges	Data or model changes Multiple, partial or grouped models	Parameter space enumeration Legal parameter combinations

Table 2: Opportunities and Challenges from User Models

4. OPPORTUNITIES AND CHALLENGES

Now that we have captured a user model and have judged its quality to be satisfactory, we can use the model in the two main areas of improving physical storage and approximate query processing. There are many opportunities and challenges that arise from using these user models, and the following discussion is by no means complete. Table 2 contains a short overview of the points discussed in this section.

4.1 Physical Storage

⊕ **"True" semantic compression** Semantic compression relies on a necessarily small number of models hard-coded into a system [5]. Compression algorithms perform best if the underlying mathematical model closely approximates the data to be compressed. If we use the user-supplied model as a compression model, we can expect high compression rates, which in turn allows more data to be stored and faster access. A straightforward compression method would be to store only the differences between the predicted and observed values. Using the model and trained parameters, we can then recompute the original dataset without loss of information.

⊕ **Zero-IO Scans** In the case of approximate queries, we do not even need to access the stored data at all, since we can use the model instead of the stored data to provide values. This allows us to transform an IO-bound problem (scanning a large table on disk) into a CPU-bound problem (recalculating all the values from the model). At the same time, we expect improved accuracy compared to database synopses.

⊖ **Data or model changes** While we expect to be able to retain models forever, the user might supply us with even better models in quick succession. If we base our data compression on a model, we can choose to recompress the data, which is an IO-intensive process. Changing or added observations can change fit of the model dramatically. This could also make a model with a previously poor fit relevant again. A possible solution could be to check these measures for all previous models and switch when appropriate.

⊖ **Multiple, partial or grouped models** First, there might be multiple such models that are of high quality and that involve equivalent or overlapping sets of parameters and output values. It is not obvious how to select the best model. Second, it is far from certain that a model covers complete columns in a table. For example, if the model has been fit on a query result that restricted the tuples, the model and its fitting parameters are only applicable to this subset. For selections that overlap both the modeled and the "unmodeled" area, it is not obvious whether the model should be involved at all. Third, models might be fitted to results of aggregation queries. For example, in our LOFAR example, we have seen how a model may be only applicable to a single group. Even if a single model is used, we would get a set of model parameters for each aggregation group. How this multi-parameter model is to be mapped back to a source table is an open issue to say the least.

4.2 Approximate Queries

⊕ **Analytic solutions for linear models** One of the greatest opportunities in capturing user models is their possible use for quick approximate query answering. Previous work has already discussed how histograms can be used for approximation of query answers [9]. For the common class of linear models, we can even go one step further and calculate analytic solutions for aggregation queries. For example, given a well-fitting linear model we can calculate the minimum and maximum value for a column.

⊕ **Model exploration** We can facilitate the exploration of the model's domain by the user. For example, we can find interesting subsets of the data by analyzing the first derivative of the model function for regions in the parameter space with high gradients.

⊕ **Data anomalies** Often, the observations that do not fit the model are of supreme interest. These will stand out in the fitting process by for example showing large residual errors. This information could be used for data exploration, in particular when calculating the "interestingness" of subsets of the data [15]. In our LOFAR example, there is a small number of radio sources where the intensity is seemingly unrelated to the frequency. Obviously, these are of interest to further study.

⊖ **Parameter space enumeration** Recall our astronomy example, we know that the model approximates I given ν and a source identifier S with associated p and α constants. But what if one or both of these parameters are not specified in the query? We could load the missing parameters from the measurement data, but the cost for this could quickly overwhelm the savings. In the case of missing S, while we could regenerate p and α using the raw data, we might as well use the raw data directly. However, if a parameter column is enumerable, we can use it without actually loading its values. Straightforward examples for enumerable columns could be continuous integer timestamps, as they appear for example in tables containing time series. Similarly, categorical variables can be replaced by a small set with all the values they assume. To stay in the example, our telescope only creates observations at a small set of frequencies, so ν would only assume values in $\{0.12, 0.15, 0.16, 0.18\}$. It is therefore possible that we can enumerate all possible combinations. Then, we can apply the model to each combination.

The main problem here is to coerce a continuous model that can generate an infinite amount of data points into the relational schema. In previous work, this is referred to as griding [7, 19]. However, in our concept, the model merely acts as an auxiliary data structure to speed up approximate queries or improve storage. Hence, the dimensions and values of the inputs are known.

⊖ **Legal parameter combinations** An interesting question also concerns point queries for which no data is present in the original dataset. It is far from certain that all possible combinations of input parameters were part of the original table. In this case we would violate relational semantics due to additional results that were not in the original data set. There are two possible solutions to the issue: First, we could require the model implementation to restrict the *legal values of the parameter space*, for example by supplying a filter function. Second, we could generate a compressed lookup structure (e.g. Bloom filters) to encode all legal parameter combinations.

5. RELATED WORK

With regards to related work, there are three very relevant previous approaches. Deshpande and Madden presented MauveDB [7], which proposes an abstraction called "model-based views". These views are able to abstract from the underlying measurement data based on a user-specified model. The system was developed to support distributed data collection from sensor networks. The parameter space enumeration issue is avoided by projecting the raw data onto a grid with fixed boundaries. This way, the number of data points generated from the model is fixed, which fits well with the relational model. However, the models that underlie these views have to be explicitly implemented in MauveDB, and users have to make a conscious and explicit decision to use this feature for their data. The paper merely discusses regression and interpolation as possible models. What we propose here is a far more flexible approach, where models appear as a by-product from in-database statistical analyses. Subsequent work by Thiagarajan and Madden (FunctionDB) extends the concept with an algebraic query processor [19]. The functions that are fitted to the data are piecewise polynomial functions. The system also contains an optimizer which avoids grid materialization as long as possible for improved performance.

On the storage optimization side, Babu et al. proposed the SPARTAN system, which uses model-based compression to reduce storage costs and increase access speed [5]. The system uses Bayesian networks to detect possible data dependencies and then trains a decision tree to predict the dependent variables. This tree is then used to replace the dependent variables in compressed tables. However, even though the compression is lossy, the system is only barely able to outperform standard gzip compression.

Akdere et al. [3] argue that data management should also encompass model management. There, models are used both for self-management such as query optimization as well as predictive data analysis. They describe the Longview database system design and architecture, where a model management component constantly trains models (based on a fixed set of model templates) for predictive purposes.

Zimmer et al. [22] investigate the use of continuous models in the data management context. They focus particularly on inverse prediction. Given a model and desired output, they search for the input values that are likely to create this output. They use two distinct approaches: First, Inverse Regression, where the roles of the input and output variables are swapped and a new regression model is trained. Second, Restraint Optimization, which follows a geometric intuition where the input space is restricted to only allow the space that describes the possible values for the input variables. They also show how complex relational-style queries can be formulated and answered on these models. Range and point queries are easily translated into geometric restrictions for both output and input variables. This also addresses MauveDB's and FunctionDB's issue of limiting the parameter space through gridded input for linear regression models. However, there is no free choice of model and their optimizations are not easily translated to the non-linear case.

6. CONCLUDING REMARKS

In this paper, we have argued that statistical models should no longer be ignored by data management. These models are highly valuable for both storage organization as well as for the computation of fast approximate answers to analytical queries. We propose to intercept the model fitting process by using the existing integrations of statistical facilities into relational databases [14, 11]. To the best of our knowledge, such a system has neither been proposed nor implemented yet. However, considerable future work is needed to make user models an integral part of data management systems. There are numerous conceptional and practical issues to overcome, some of which we have discussed here.

For the next steps, we propose to create a proof-of-principle implementation of the first step in the process, that captures the user model as it is being fitted. A straightforward way of evaluating this system would be to create models that describe the considerable regularity in the generated datasets for popular database benchmarks such as TPC-DS. Then, the complex benchmark queries serve as tasks for approximate query answering, while the data itself provides a playing field for model-based storage optimizations such as semantic compression.

We also suspect that focusing on a single class of models as previous work has [7, 19, 22] is unlikely to cover enough ground. Another area of future work would thus be to survey scientific fields and their models, in order to gain a better understanding of the complexity involved. However, databases face a challenging task in executing arbitrary user code as part of the query. Optimizer decisions are usually based on a precise understanding for example of the dimensionality of operator results. This cannot be assumed to be present for user models, and forcing users to annotate their models accordingly is unlikely to become popular. A learning query processor might be the only sensible choice here [17].

We hope to have conveyed our enthusiasm for moving the scientific process closer to the data. This would allow data management systems to benefit from the experience and domain knowledge of their users, which describe the "laws of data nature".

Acknowledgments
We thank Bart Scheers for providing the LOFAR example data and explanations. This research is supported by NWO through the `COMMIT/` project.

7. REFERENCES

[1] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. Join synopses for approximate query answering. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, pages 275–286, New York, NY, USA, 1999. ACM.

[2] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: Queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, pages 29–42, New York, NY, USA, 2013. ACM.

[3] M. Akdere, U. Çetintemel, M. Riondato, E. Upfal, and S. B. Zdonik. The case for predictive database systems: Opportunities and challenges. In *Proceedings of the Biennial Conference on Innovative Data Systems Research*, CIDR '11, pages 167–174, 2011.

[4] I. Alagiannis, S. Idreos, and A. Ailamaki. H2o: A hands-free adaptive store. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 1103–1114, New York, NY, USA, 2014. ACM.

[5] S. Babu, M. Garofalakis, and R. Rastogi. Spartan: A model-based semantic compression system for massive data tables. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, pages 283–294, New York, NY, USA, 2001. ACM.

[6] B. Burke and F. Graham-Smith. *An Introduction to Radio Astronomy*. Cambridge University Press, 2010.

[7] A. Deshpande and S. Madden. Mauvedb: Supporting model-based user views in database systems. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, pages 73–84, New York, NY, USA, 2006. ACM.

[8] P. B. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 331–342, New York, NY, USA, 1998. ACM.

[9] Y. E. Ioannidis and V. Poosala. Histogram-based approximation of set-valued query-answers. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 174–185, San Francisco, CA, USA, 1999. Morgan Kaufmann.

[10] H. Mühleisen and T. Lumley. Best of both worlds: Relational databases and statistics. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, SSDBM, pages 32:1–32:4, New York, NY, USA, 2013. ACM.

[11] Oracle. *Bringing R to the Enterprise*. `http://www.oracle.com/technetwork/database/options/advanced-analytics/r-enterprise/bringing-r-to-the-enterprise-1956618.pdf`, accessed 2014-10-19.

[12] S. Psillos. The cognitive interplay between theories and models: The case of 19th century optics. *Theories and Models in Scientific Processes, Poznan Studies in the Philosophy of the Sciences and the Humanities*, (44):105–133, 1995.

[13] M. Redhead. Models in Physics. *The British Journal for the Philosophy of Science*, 31(2):145–163, 1980.

[14] SAP. *SAP HANA R Integration Guide*. `https://help.sap.com/hana/SAP_HANA_R_Integration_Guide_en.pdf`, accessed 2014-10-19.

[15] T. Sellam and M. L. Kersten. Meet charles, big data query advisor. In *Proceedings of the Biennial Conference on Innovative Data Systems Research*, CIDR '13, 2013.

[16] E. Sidirourgos, M. L. Kersten, and P. A. Boncz. Sciborq: Scientific data management with bounds on runtime and quality. In *Proceedings of the Biennial Conference on Innovative Data Systems Research*, CIDR '11, pages 296–301, 2011.

[17] M. Stillger, G. Lohman, V. Markl, and M. Kandil. LEO - DB2's learning optimizer. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 19–28, Orlando, Sept. 2001. Morgan Kaufmann.

[18] T. Strutz. *Data Fitting and Uncertainty: A Practical Introduction to Weighted Least Squares and Beyond*. Vieweg+Teubner Verlag, 2010.

[19] A. Thiagarajan and S. Madden. Querying continuous functions in a database system. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 791–804, New York, NY, USA, 2008. ACM.

[20] M. P. van Haarlem et al. LOFAR: The LOw-Frequency ARray. *Astronomy & Astrophysics*, 556, Aug. 2013.

[21] J. Wolberg. *Data Analysis Using the Method of Least Squares: Extracting the Most Information from Experiments*. Springer, 2006.

[22] A. M. Zimmer, P. Driessen, P. Kranen, and T. Seidl. Inverse predictions on continuous models in scientific databases. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*, SSDBM '14, pages 26:1–26:12, New York, NY, USA, 2014. ACM.