Transparent Forecasting Strategies in Database Management Systems

Ulrike Fischer(✉) and Wolfgang Lehner

Technische Universität Dresden, Database Technology Group, Dresden, Germany
{ulrike.fischer,wolfgang.lehner}@tu-dresden.de

Abstract. Whereas traditional data warehouse systems assume that data is complete or has been carefully preprocessed, increasingly more data is imprecise, incomplete, and inconsistent. This is especially true in the context of big data, where massive amount of data arrives continuously in real-time from vast data sources. Nevertheless, modern data analysis involves sophisticated statistical algorithm that go well beyond traditional BI and, additionally, is increasingly performed by non-expert users. Both trends require transparent data mining techniques that efficiently handle missing data and present a complete view of the database to the user. Time series forecasting estimates future, not yet available, data of a time series and represents one way of dealing with missing data. Moreover, it enables queries that retrieve a view of the database at any point in time — past, present, and future. This article presents an overview of forecasting techniques in database management systems. After discussing possible application areas for time series forecasting, we give a short mathematical background of the main forecasting concepts. We then outline various general strategies of integrating time series forecasting inside a database and discuss some individual techniques from the database community. We conclude this article by introducing a novel forecasting-enabled database management architecture that natively and transparently integrates forecast models.

1 Introduction

We can observe the transition of traditional data warehouse systems to big data stores where massive amount of data arrives continuously in real-time from vast data sources. Whereas traditional systems assume that data is complete or has been carefully preprocessed using ETL tools, this is not true any more in the context of big data and real-time requirements. Nowadays, data is increasingly characterized by incompleteness, inconsistency, and imprecision. Nevertheless, we can observe the requirements for sophisticated adhoc queries that extract higher-level information out of the vast and incomplete data sets. This leads to tremendous challenges of dealing with missing (past, present, and future) data that arrives at a later point in time or might even never be available.

In fact, we require data mining techniques that fill in such non-existent data. Various techniques from the statistical field aim at dealing with different kinds of incomplete data, for example:

E. Zimányi (Ed.): eBISS 2013, LNBIP 172, pp. 150–181, 2014.
DOI: 10.1007/978-3-319-05461-2_5,

- *Forecasting* estimates future, not yet available, data of a time series.
- *Imputation* replaces missing data with substituted values based on similar values observed in the same data set.
- *Interpolation* estimates a function between known data points to construct new data points.
- *Extrapolation* estimates the characteristics of a whole population based on the selection of a subset of individuals.
- *Recommendation* provides user ratings of items that have not been rated yet.

In the past, these techniques were often performed by highly qualified statistical experts, with long experience in the company, who manually experiment with different algorithms and parametrization. Such manual approaches are infeasible or even impossible for large data sets that grow and evolve at a rapid pace. Moreover, the non-expert user does not care about advanced data mining techniques. The user expects a complete view of the data set at any point in time, independent of its actual characteristics. Along with the traditional ANSI/SPARC architecture of a DBMS, which has the goal to separate a users' view of the data from its physical representation, incomplete data should be handled transparently to the user by the underlying database system. As a consequence, data mining techniques dealing with incomplete data should be seamlessly integrated into the existing infrastructure of a database management system, maintaining the declarative interface of a DBMS.

For example, assume a market research company that collects sales data according to various retailers and product lines. Retailers communicate their data to the market research company in regular time intervals. In this context, it happens quite regularly that retailers fail to deliver data, provide incomplete data or delay data delivery. Suppose a decision manager wants to create an aggregated report of products sold between yesterday and tomorrow:

```
SELECT pname, time, SUM(salesunits)
FROM facts f, products p
WHERE f.product_id = p.product_id
AND time in (yesterday(), tomorrow())
GROUP BY pname, time
```

Within this time interval, some stores might have failed to deliver their sales data, so here an interpolation model could fill in such missing data. Alternatively or additionally, an imputation model can be used to derive missing data from similar stores. Finally, the data for tomorrow is not available at all, therefore, forecast models can be used to estimate future data. All these models are created and transparently processed by the query engine, the user just receives a traditional relational table containing the complete result set.

In fact, as most mining techniques are based on some kind of statistical model, this will lead to a *model-based database system*, where incomplete, inconsistent or imprecise data is represented trough statistical models. New query processing, optimization, and execution techniques are required that work with models instead of real data. Besides query processing, such a model-based database system also impacts the design of a database to select and parametrize models that

allow efficient query processing as well as accurate query results. Finally, new data has to be efficiently incorporated into the existing model configuration.

The design of such a model-based database system opens up many challenges. We have to deal with large amount of data incorporating real-time data streams from many data sources. There are loads of existing statistical models, ranging from very simple to rather complex ones with many parameter and tuning opportunities. Both aspects, data size and parameter possibilities, pose high challenges to the design of such a model-based database system. In addition, data characteristics as well as query workloads are rapidly changing requiring a self-adaptive and self-tuning approach.

Along with this overall goal of designing a model-based database system, in this article, we turn our attention to one specific statistical technique, which is time series forecasting. Hence, we are only interested in providing future, not yet available data of a time series. Hereby, we focus on integrating *forecast models* into a database management system in order to *transparently* support queries on a future time interval, i.e., *forecast queries*.

The remainder of this article is organized as follows: We first outline the main challenges of time series forecasting using three application examples (Sect. 2). We continue by discussing the mathematical foundations of forecasting, including frequently used statistical methods in this area (Sect. 3). We then dive into technical aspects of extending database systems and give first of all a high level overview of integrating statistical methods, but not necessarily time series forecasting, into database management systems (Sect. 4). Subsequently, in Sect. 5, we review specific database techniques that explicitly address forecasting of time series data. Based on the discussion of existing work, in Sect. 6, we introduce a novel forecasting-enabled database management system that aims to fully and transparently integrate time series forecasting within a DBMS. We finally conclude in Sect. 7 and outline some further research challenges.

2 Forecasting Applications

Time series data appears in numerous domains and often forecasting of such data is required for planning and decision making processes. In this section, we discuss the characteristics of time series forecasting on three selected application areas, namely production planning, energy load balancing, and online display advertisement.

2.1 Production Planning

Typically, large volumes of historical sales data is stored in data warehouse systems and collected according to various characteristics of products, stores, and customers. Often a multidimensional data model is used for such kind of applications where different facts (often called measures) are organized along several dimensions [23]. The measures within a dimension are further divided into hierarchies to support multiple granularities. In this context, analytical database

queries are not interested in single measures but in some form of summarized data (e.g., sales in a certain area). The dimension hierarchies provide the key navigation paths for interactive OLAP (Online Analytical Processing) on the data, allowing for meaningful query formulation via drill-down, roll-up, or slice-and-dice operations [47]. Besides querying historical data, forecasts of sales figures form the basis for planning in many commercial decision-making-processes in logistics and supply chain management.

According to Mentzer and Bienstock [57], a sales forecasting system should follow several principles. First, sales forecasts should be provided in a central system and tightly coupled with the database management system, allowing fast access by various departments, such as production, distribution, and marketing. Second, forecasts have to be available for various horizons (short-, mid-, and long-term) and hierarchical levels, depending on the company's needs. For example, supply chain managers require long-term sales forecast to plan production and storage facilities, whereas short-term forecasts are required for timely transportation decisions. Third, the complexity of forecasting should be hidden from a decision manager, who is usually not an expert in the statistical area. Forecast results need to be provided in an easy-to-use format. Fourth, a sales forecasting system should include a suite of time series techniques and provide a combination of different techniques to benefit from their specific advantages, where Mentzer and Bienstock see time series, regression and qualitative techniques (see Sect. 3) as the most important approaches in sales forecasting. Finally, the best forecasting techniques for a time series should be selected automatically, by trying a number of different techniques and selecting the technique that provides the best forecast accuracy.

2.2 Energy Load Balancing

As another example, consider the energy market domain. One major challenge is the constantly increasing capacities of renewable energy sources (RES) due to governmental regulation efforts (e.g., climate saving propositions) and excessive funding policies [21]. Renewable energy sources pose the challenge that production depends on external factors (wind speed, amount of sunlight, etc.). Hence, available power can only be predicted but not planned, which makes it rather difficult for energy distributors to efficiently include RES into their daily schedules.

The key to balance an energy distribution network successfully is to predict as many of the most influencing (correlated) parameters for operations as possible. Such forecasts are often made by domain-specific forecasting techniques specifically designed for energy demand or supply. To forecast energy demand, for example, Ramanathan et al. [72] propose a multi-equation forecasting technique that creates a different statistical model for each hour of a day and includes various variables that capture the seasonality of the data as well as external influences (e.g., temperature).

In the past, energy balancing was typically done once per day at a specific time and, accordingly, one-day ahead demand forecasts were calculated only on

a daily basis. The need for fast response times to react to new market situations (e.g., weather changes) as well as the continuous streams of new demand and supply measurements poses additional real-time demands on the forecasting process [26]. Thus, the runtime of forecasting is very critical and, more importantly, forecast models have to be continuously adapted to changes in the time series behavior. Moreover, the hierarchical organization of the energy market requires a careful selection of the forecasting granularity (e.g., single wind installation vs. complete regions), the efficient handling of real-time mass prediction processes and the guarantee of consistency between hierarchy levels.

2.3 Online Display Advertisement

As a third example, online display advertisement allows advertisers to promote products to users by having publishers display their graphical ads on web pages. For example, a brokerage firm may wish to target males from California who visit a Finance web site, and show an ad promoting its special offers to those users. Such kind of targeted ads are channeled to users via ad networks — intermediates that package and sell ad space from multiple publishers' websites [82]. In order to be able to accept contracts and allocate inventory, an ad network has to have access to reliable forecasts of user visits. Overestimating user visit volumes may result in penalties for the publisher if guarantees are not met, whereas underestimating user visits may leave unsold user visits that often result in substantial revenue loss.

The forecasting problem in online display advertisement has several challenges [1]. First, the data to be forecasted is very high-dimensional. Specifically, each user visit is characterized by hundreds of attributes, including the demographics of the user (e.g., age, gender, location), explicitly stated interests of the user (e.g., travel, spots), implicitly inferred interests of the user (e.g, planning a vacation), characteristics of the web page being visits (e.g., sports page, travel page), and characteristics of the system being used by the user (e.g., PC vs. mobile, IP address location). Second, as a consequence of the high-dimensionality of the data, the number of combinations that needs to be forecasted is of the order of trillions. A forecast can be requested for any combination of the hundreds of attributes using arbitrary forecasting methods, ranging from traditional time-series forecasting techniques up to latent class models [12]. Nevertheless, forecasts have to be returned in real-time, of the order of a few hundred milliseconds. Several queries are issued to the forecasting system within a short span of time to decide if an advertisers' contract should be acceptable.

3 Mathematical Foundations of Time Series Forecasting

All three application areas are based on the traditional model-based time series forecasting process, which we outline in this section. After introducing the basic idea and terminology of forecasting (Subsect. 3.1), we detail the three main steps of forecasting, namely model creation (Subsect. 3.3), model usage (Subsect. 3.4),

and model maintenance (Subsect. 3.5). For further readings we refer to standard literature about time series analysis and forecasting [9,10,13].

3.1 Basic Idea and Terminology

A *time series* is sequence of observations taken sequentially in time, spaced at *equidistant* time intervals. A time series up to the *current time* t is denoted as:

$$X = (x_1, x_2, ..., x_t). \tag{1}$$

In general, each point in time might be associated with multiple observations, where we distinguished *dependent* and *independent* observations or variables. Dependent variables, also known as *measure* or *output* variables, are those variables we actually want to forecast (e.g., product sales, solar energy production). In contrast, those variables that we believe influence the value of the dependent variables are referred to as independent or explanatory variables (e.g., product price, sun radiation).

Forecasting refers to the estimation of values of a time series at some future point in time. These future values are called *forecast values* or short *forecasts* $\hat{x}_{t+k}$. The forecasts $\hat{x}_{[t+1;t+h]}$ in the interval from the next point in time $t+1$ up to time $t+h$ are usually of most interest. The length of the interval is denoted as *forecast horizon* h. Note that the term *prediction* is often used in a more general sense and covers different problem types, e.g., classification, recommendation or moving object prediction. In contrast, the term forecasting is only concerned with estimating the next and future values of a sequence, i.e., a time series.

The quality of forecast values can be expressed by calculating *prediction intervals*, which, for a certain probability, give an estimate of the interval in which forecast values will fall. If new real data is available, the error of the forecasts can be calculated by comparing forecasts with real time series data using an error metric, e.g., the *mean squared error* (MSE).

A *forecasting method* is a procedure for computing forecasts from present and past values. Most forecasting methods are based on a *forecast model*, which is learned over historical training data and used to compute the forecast values (*model-based forecasting*). Examples of approaches that do not belong to this class of methods are judgmental or similarity-based forecasting methods.

A *forecast model* consists of the following parts:

- definition of *input* and *output* time series,
- definition of the *forecasting method*, which determines how forecast values are calculated,
- *model parameters* of the forecasting method that have to be determined in the model estimation step, and
- *model state*, representing internal variables of the forecasting method that change with time.

The *input* of a model consists of n dependent and m independent variables, whereas the *output* yields from the associated forecasts of the n dependent variables. The *model parameters* and *model state* directly depend on the used forecasting method.

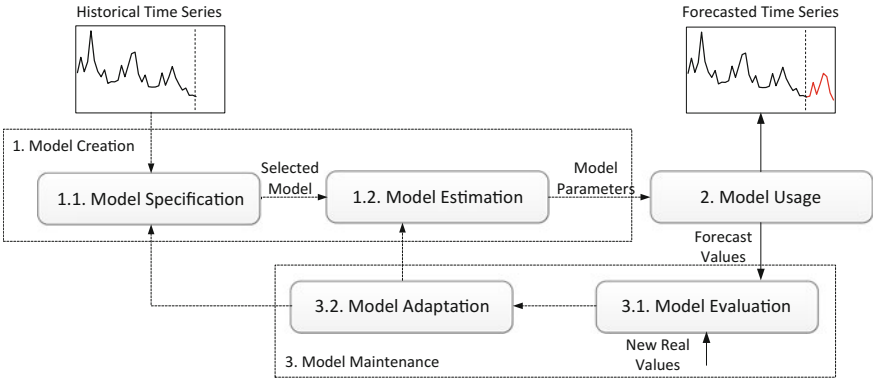

Fig. 1. General forecasting process

Finally, the different steps of forecasting can be summarized into a general *model-based time series forecasting process* (Fig. 1):

1. *Model Creation:* A forecast model is created by defining input, output as well as the forecasting method (*model specification*), and by estimating the model parameters (*model estimation*).
2. *Model Usage:* Forecast values based on the created forecast model are calculated.
3. *Model Maintenance:* The forecast model is evaluated by comparing real time series data with forecast values (*model evaluation*) and, optionally, *model adaption* is triggered by recalculating the model parameters or choosing a new model.

In what follows, we first discuss the characteristics of different forecasting methods and describe two of them in more detail. We then have a closer look at the three main steps of the forecasting process.

3.2 Overview Forecasting Methods

Numerous forecasting methods have been proposed in the literature, e.g., Gooijera and Hyndman [36] summarized over 940 papers in the period 1982–2005. Forecasting methods are often classified into *time series methods* (or univariate methods) and *causal methods* (or multivariate methods) [13]. Time series methods assume that forecasts depend only on present and past values of the single series being forecasted, possibly augmented by a function of time such as linear trend. In contrast, in causal methods forecasts depend, at least partly, on one or more additional (independent) variables (e.g., price, weather). Finally, forecasting can also be implemented by *machine learning approaches* that exploit artificial intelligence techniques, e.g., neural networks [8]. Machine learning approaches are not specifically designed for time series data and might be applied for

Table 1. Classification of forecasting methods

Time series methods	Causal methods	Machine learning
ARIMA [9]	Multivariate linear	Neural networks [88]
Exponential	Regression [86]	Support vector
Smoothing [40]	ARMAX [9]	Machines [60]
Multiple linear		Bayesian networks [87]
Regression [34]		Decision trees [56]

arbitrary predictive tasks, e.g., classification. Table 1 summarizes the different forecasting methods and gives some literature examples where these methods have been applied for time series forecasting. Many extensions to the basic formulation of those forecasting methods have been developed, including domain specific methods that are specifically designed to solve a prediction task in a certain domain.

Various studies have compared the accuracy of the different forecastingarbitrary methods with varying results depending on the domain and forecasting target. Exponential smoothing and ARIMA models have shown empirically to be able to model a wide range of real world time series [54], and are usually computationally more efficient than elaborate machine learning approaches. The main idea of both approaches is sketched in the following.

Exponential Smoothing. Exponential Smoothing is a popular scheme to produce smoothed time series, where past observations are weighted with exponentially decreasing weights [33]. In other words, recent observations are given more weight in forecasting than older observations. Different variants of exponential smoothing have been proposed, varying in the number and characteristics of the smoothing weights. The most common variants are called *single*, *double*, and *triple* exponential smoothing.

For example, single exponential smoothing has only one weight parameter, also known as the smoothing constant α:

$$a_t = \alpha x_t + (1 - \alpha) a_{t-1} \qquad \text{with } a_0 = x_0. \tag{2}$$

The forecast of single exponential smoothing is a constant value based on the last smoothed value a_t, independent of the forecast horizon h:

$$\hat{x}_{t+h} = a_t. \tag{3}$$

This method is mainly used for stationary time series fluctuating around a constant mean. In contrast, double exponential smoothing introduces an additional trend component, whereas triple exponential smoothing (also known as Holt-Winters [40]) further applies a seasonal component.

ARIMA Models. ARIMA models describe the behavior of time series using an auto-regression process [9] and consist of three main parts; the autoregression part (AR), the integration part (I) and the moving average part (MA).

The autoregressive part $AR(p)$ is computed by a linear combination of previous values up to a defined maximum lag (denoted p) combined with a random error term ϵ_t:

$$x_t = \sum_{i}^{p} \phi_i x_{t-i} + \epsilon_t. \tag{4}$$

In contrast, the moving average part $MA(q)$ describes a time series as a random error term plus some linear combination of previous random error terms up to a defined maximum lag (denoted q):

$$x_t = \sum_{i}^{q} \phi_i \epsilon_{t-i} + \epsilon_t. \tag{5}$$

Hereby, the random error terms result from a white noise process, i.e., a set of uncorrelated, normal-distributed, random variables with an assumed equal variance. Most time series can not be described solely by an AR or MA forecast model, as they show behavior of both models at the same time. For this reason, a combination of both models is useful. Additionally, an integrated part I adjusts the model for non-stationary time series, leading to so called $ARIMA(p, d, q)$ models. This basic formulation of ARIMA models can be extended to include seasonality (SARIMA) or exogenous data (ARIMAX).

A variety of research has studied the relationship between exponential smoothing and ARIMA models [33,59]. In general, all linear exponential smoothing methods have equivalent ARIMA models. However, exponential smoothing is often preferred over ARIMA due to its simplicity, robustness and the surprising accuracy that can be obtained with minimal effort in model selection.

3.3 Model Creation

Model creation is the process of defining (model specification) and training (model estimation) a forecast model for given time series data. In this section, we discuss each of these steps separately. However, in automatic model identification approaches these two steps are often combined as models are built and evaluated iteratively.

Model Specification. Model specification requires the definition of the input and output time series as well as the definition of the forecasting method.

Input and Output Selection. The output time series (i.e., the dependent variables) depends on the aim of forecasting and has to be specified by the application, whereas the input time series (i.e., the independent variables) might be manually or automatically selected. Hereby, the goal is to find those variables (often called features) that are highly correlated and significantly contribute to the time series

to be forecasted. Statistical techniques, such as principal component analysis, are often applied to find relationships between different features and to reduce the number of features [32]. Another issue in input selection is given by the time series history, e.g., how much history should be used for model estimation. Hereby, existing research has studied the minimal historical length required for specific forecasting methods [43] as well as the influence of the history length on the forecast accuracy [5] and the runtime of the parameter estimator [77].

Forecasting Method Selection. For a given data set, we need to select the forecasting method with the highest accuracy, which can be done manually or automatically. The manual approach requires knowledge of statistical theory and uses diagnostic tools such as the correlogram. Choosing a model manually is not possible if a large number of time series is involved or if the forecast is done by non-experts in the statistical area. In the automatic approach, the best model is selected empirically according to the in-sample error or a model selection criteria such as the Akaike's Information Criterion (AIC). AIC chooses the model that maximizes the so-called likelihood function and includes a regularization term, which basically avoids overfitting by increasing the training error with the number of parameters fitted in the model. Hyndman et al. [42] developed a state space framework for the class of exponential smoothing methods, where the best method is chosen automatically based on AIC. Heuristic model identification approaches for the class of ARIMA models have also been developed [41]. All automatic approaches still have the drawback that they select a single model that has to be best at all times. Ensemble approaches increase robustness by combining forecasts of several models using weighted linear combinations [36].

Model Estimation. In model estimation, the parameters of the forecasting method (called *model parameters*) are fitted to a given training time series. Thus, model estimation tries to find the best parameter combination for the training data. This process involves two main components — an *optimization function* (to specify which parameter combination is best) and an *optimization approach* (to control the search strategy). Most common optimization approaches follow an iterative search process based on the steepest descent or hill climbing technique. In each iteration one or several parameter combinations are evaluated using the optimization function and subsequently, based on the outcome of the evaluation, a new parameter combination is chosen for the next iteration. After termination, the best parameter combination according to the optimization function is outputted (Fig. 2).

The optimization function is composed of the forecasting method and the error metric used to evaluate the forecast values. Commonly used error metrics are least squares or maximum likelihood approaches.

Optimization approaches are mainly distinguished into derivative-based and derivative-free algorithms. Derivative-based methods (e.g., gradient descent, quasi-newton) exploit the optimization function's first or second derivations to move directly into the direction of the steepest descent. If the optimization function is not derivable gradient approximation techniques can be applied.

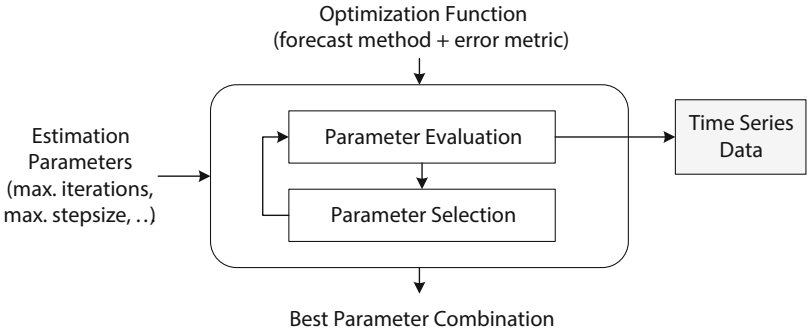

Fig. 2. Parameter estimation process

In contrast, derivative-free algorithms (e.g., simulated annealing, nelder-mead) make only direct evaluations of the optimization function, i.e., treat it as black-box. Finally, the optimization approach can be configured with various parameters, such as the maximum number of iterations or the maximum step size for selecting the next parameter combination.

3.4 Model Usage

Model usage applies the created model to forecast future values of the time series for a given forecast horizon h. In model-based forecasting, the time-consuming part is given by the model creation step, whereas model usage requires only the application of a function (with the trained parameters).

A more complex aspect of model usage concerns the aggregation of time series data. Time series data may be aggregated either across time, called *temporal aggregation*, or across several time series, called *contemporaneous aggregation* [13]. For example, suppose we have sales figures for different brand sizes of different products in successive weeks. Such data may be quite volatile and difficult to forecast without some form of aggregation, either across time (e.g. over successive 4-week periods) or across products (e.g. sum all brand sizes for the same brand). A common problem in inventory control is whether to develop a summary forecast for the aggregate of a particular group of items and then allocate this forecast to individual items based on their historical relative frequency, called the *top-down approach*, or make individual forecasts for each item, called a *bottom-up approach*. This line of research is called *hierarchical forecasting* [31].

3.5 Model Maintenance

As time proceeds, new values of the time series are observable, which impact the forecast model. First, the state of the model has to be updated to the current time series values, which we refer to as *model update*. Second, the parameters of the model or even the forecasting method might change, which is meant by

the term *model maintenance*. Complex seasonal patterns or unexpected changes in time series' characteristics (also called concept drift) like customers' buying preferences or the influence of weather predictions usually require such an adaption of the model. Model maintenance exhibits two major challenges: (1) when to trigger forecast model maintenance (model evaluation) and (2) how to efficiently adapt the forecast model parameters (model adaption).

Model Evaluation. If new real data is available, the model can be evaluated by calculating the forecast error using a specified error metric. Commonly known error metrics are the mean absolute error (MAE) and mean squared error (MSE). Such error metrics, however, depend on the scale of the time series values, are hard to judge and do not allow comparisons between time series of different scale or mean. In contrast, percentage error metrics evaluate the error's magnitude instead of its size. One example is the *symmetric mean absolute percentage error* (SMAPE) [53], which was also used in the M3-competition — a mayor time series competitions in the business forecasting domain [54].

Simple model evaluation approaches trigger model adaption independently of the actual time series data and might be time-based (after a time interval), update-based (after a fixed number of new values) or event-based (after updating an exogenous variable or on request). More advanced approaches monitor the temporal development of the time series. For example, error-based approaches evaluate the forecast error after each new real value and trigger adaption whenever the error surpasses a predefined threshold [16]. Other approaches use statistical information about the time series like minimum and maximum values [39] or statistical tests [51].

Model Adaption. A simple way to realize model adaption is by starting from scratch and by re-executing the model estimation and, optionally, the model identification step as done in the initialization. This can easily be improved by reusing previous information, e.g., by providing the last parameters of the model as starting parameters to the model estimation step. A more advanced approach stores previous model parameters in a decision tree according to specific context information (e.g., type of day, temperature) and uses them as starting point in the estimation step if the same context reoccurs [17]. Another approach extends genetic algorithms with dynamical features, where previously good models are given an advantage in future selection rounds [83]. Orthogonal approaches adapt the training set of the models and include only recent observations in the model creating step, either employing fixed-size or dynamic windows [85]. All these approaches use offline parameter estimation approaches, where the parameters are fully reestimated using any of the previously discussed optimization approaches. As an alternative, approximate online optimization algorithms [90] may be applied, which evaluate the objective function after each new time series value and alter the parameter estimate made so far accordingly. However, the parameters by this approach are only approximative and will deviate from parameter estimates produced by full optimization. Finally, the forecast model

itself might be designed in an adaptive way, which aims at completely avoiding the need for recalculating the model parameters. Self-adaptive forecasting methods extend existing forecasting methods with time-dependent parameters (e.g., [72]). Ensemble methods combine forecasts of several models and adapt the weights of the ensemble members over time.

4 Architectural Integration

Recall our vision of a model-based database system outlined in the introduction. In terms of time series forecasting, our overall objective is the transparent integration of forecast models inside a database management system. We discussed the main challenges of typical forecasting applications in Sect. 2, such as high-dimensional data, real-time requirements, complex and domain-specific forecasting methods, diverse workloads, non-expert users, and fast evolving data sets. Moreover, in Sect. 3, we highlighted the most important steps of the forecasting process and outlined challenges in this context, such as the selection of the best forecasting method, long parameter (re-)estimation times, and the importance of a smart maintenance strategy.

We now turn our attention to the actual integration of forecasting inside a database management systems. First of all, in this section, we review general solutions to the integration of any kind of statistical method, not necessarily time series forecasting, into a DBMS and outline to what extent they support our overall objective as well as the discussed challenges. We classify existing methods into (1) no integration approaches (Subsect. 4.1), (2) partial integration approaches that try to keep changes to the database as small as possible (Subsect. 4.2), and (3) full integration approaches that actually extend the functionality of a database system (Subsect. 4.3).

4.1 No Database Integration

No database integration approaches refer to the use or integration of statistical approaches in other system categories, such as external software or Map-Reduce environments.

Statistical Software Environments. Traditionally, statistical computations have been performed outside the database system by specialized software, which uses the DBMS primarily as backend data server. Ganesan and Shenoy [50] provide a survey of forecasting software up to the year 2006. Probably the most well-known commercial software environments are Matlab [55], SAS [75] and SPSS [78]. All include a large variety of specific forecasting methods including approaches for automatic and ensemble forecasting.

A popular open-source statistical software package is the R framework [71]. With over 2,000 add-on packages, it is comparable to the big commercial packages SAS and SPSS. In the context of time series forecasting, it contains a wide

variety of model types and parameter estimators. Model types range from linear models, exponential smoothing, ARIMA up to machine learning approaches. Additionally, automatic model identification approaches for ARIMA and exponential smoothing models are available [41]. A general-purpose optimization functions including five different optimization approaches (e.g., Nelder-Mead, Simulated Annealing) is used to estimate the parameters of the various model types. As R is open-source it can be easily extended with new, domain-specific, forecasting methods. A number of approaches aim at improving the handling of large amount of data in R [76]. The proposed techniques range from simple ones that require rewriting and adapting of existing scripts and functions up to more complex ones that try to adapt the R environment in a transparent manner. For example, an approach called RIOT [89] focuses on storing and querying arrays, and tries to make R more I/O efficient by introducing a new expression algebra.

Map-Reduce Environments. Massive data sets and and large clusters of machines have led to an increased interest in implementing statistical algorithms on Map-Reduce environments. Several research has investigated the implementing of scalable versions of machine learning algorithms on Map-Reduce, ranging from proprietary (e.g., [67]) to open source implementations [6]. In contrast, declarative machine learning approaches try to avoid the low-level implementations of specific algorithms on Map-Reduce. For example, SystemML [35] provides a declarative high-level language for writing machine learning algorithms, which is automatically compiled and optimized into a set of Map-Reduce jobs. Another declarative approach is MLbase [49], which proposes an optimizer that selects and dynamically adapts the choice of the learning algorithm.

All discussed approaches exhibit the general problem of being outside the database system. This might be valid in application scenarios where data is stored in external files or fits into main memory, and database characteristics such as transaction management are not required. However, if data is managed in

```
INSERT INTO YD
 SELECT i,j,
  sum((YV.val-C.val)**2)
 FROM YV,C
 WHERE YV.l = C.l
 GROUP BY i,j;
```

(a) Computing the Euclidean distance for k-means in SQL [64]

```
SELECT svm_regression(
 'input_table',
 'model_table',
 parallel,
 'kernel_func',
 verb DEFAULT false,
 eta DEFAULT 0.1,
 nu DEFAULT 0.005,
 slambda DEFAULT 0.05);
```

(b) Training SVM model in MAD [15]

```
initialize(ModelCoef *w)
{ .... }

transition(ModelCoef *w,
           Example e)
{ ...
 wx = Dot_Product(w,e.x);
 c = stepsize * e.y;
... }

terminate(ModelCoef *w)
{ .... }
```

(c) Implementation of SVM in BISMARCK [24]

Fig. 3. Exploiting SQL and UDFs

a traditional database management system those approaches have several drawbacks. They require data transfer from the database to the statistical software system and vice versa, might lead to inconsistencies between data and models and miss optimization potential such as the reuse of models by multiple queries. Surely, some or all of this functionality could be implemented in the external system. This, however, requires the re-implementation of existing concepts of the DBMS and will eventually lead to the design of a new database system outside of the actual database system.

4.2 Partial Database Integration

Partial database integration approaches try to leave the database system itself unchanged or include advanced analytical functionality by keeping the changes to the databases as small as possible. We distinguish three approaches in this area: SQL extensions and UDFs, customized functions, and bi-directional communication approaches.

Exploiting SQL and UDFs. The first set of approaches uses database query languages to express linear algebra functions or even higher-level algorithms and, thus, try to get a database system to act like a statistical software environment. Such approaches either (1) use directly SQL to implement data mining algorithms, (2) hide mining functionality behind user defined functions and provide high-level SQL extensions to interact with mining models and results, or (3) support the implementation of mining algorithms by providing low-level language extensions (Fig. 3).

First, SQL can be used to directly implement data mining algorithms, such as Bayesian classifiers [65] and clustering approaches [64]. Figure 3(a) shows an excerpt of the k-means clustering algorithm in SQL, namely the computation of the Euclidean distance between the data points and the centroids.

Second, high level language constructs for specific mining tasks have been proposed. The MAD approach [15] consists of a hierarchy of mathematical concepts in SQL that enable vector and matrix operations, simple functions as well as sophisticated analytical methods such as ordinary least squares, conjugate gradient, or support vector machines (Fig. 3(b)). The Splash system [22] views statistical models, such as probability density functions, as SQL aggregation operations and proposes extensions to the relational data model and SQL query language for interaction with such models. Ordonez and Pitchaimalai [66] propose a general system that integrates statistical models such as correlation, linear regression, principal component analysis, and clustering into a database using SQL queries and UDFs. Besides these general approaches, a large number of research papers has addressed specific data mining methods. Examples include association rule mining [45] and sequential patterns [74]. All approaches provide a high-level query language that hide statistical details from less sophisticated users.

In contrast, the ATLaS system [84] introduces a lower-level language, where the user can integrate simple data mining algorithms with user-defined aggregates by implementing three standard functions in SQL — initialize, iterate, and terminate. The ATLaS language processor optimizes and translates ATLaS SQL programs (e.g., decision tree classifier, apriori) into C++ code. In another work, Feng et al. [24] propose a unified architecture (called BISMARCK) for convex programming problems, such as support vector machines, where local solutions are always globally optimal. Their main component is an in-RDBMS implementation of the incremental gradient descent optimization approach that allows to solve a number of convex programming tasks in a unified way. Analogue to ATLaS, a developer can integrate analytic tasks by implementing three standard functions using user-defined aggregates, using any language supported by the DBMS (Fig. 3(c)). Additionally, performance optimizations, namely partitioning and parallelization schemes, are studied.

Exploiting SQL and UDFs for data mining algorithms has the advantage of being flexible: the analyst is able to develop algorithms independently on top of the database and nevertheless is able to profit from performance gains by running analytical methods inside the database [15,66]. However, SQL itself is not designed to express statistical computations. SQL follows a declarative logic (e.g., it lacks a convenient syntax for iteration), whereas statistical computing requires imperative and functional programming logic. This leads to a high overhead of statistical computations and makes it impossible to express sophisticated time series methods in SQL. In contrast, UDFs allow arbitrary programming languages supported by the DBMS and might be used to implement advanced time series methods. However, within all approaches models are explicitly queried and not transparently processed as first class citizens inside the database. Furthermore, within an UDF, all decision have to be made locally, whereas a DBMS exhibits a global view over all queries and operators, allowing joint optimization techniques such as model reuse and maintenance in multidimensional data sets.

Customized Functions with Proprietary Languages. Instead of developing SQL extensions, another possible approach is to implement data mining functionality internally as customized black box functions and offer proprietary languages to the corresponding methods. This approach has been used by most commercial database management systems, which provide advanced time series forecasting methods to some extend.

Microsoft SQL Server offers a Data Mining Extension (DMX) for creating models for various mining tasks such as association rule mining, clustering, and also time series forecasting [79]. Two explicit time series forecasting methods are included, autoregressive trees and ARIMA models, which are by default combined to a hybrid forecasting method. DMX supports a set of functions that allow to query a forecast model for predictions and additional statistical information (Fig. 4 (a)). Chaudhuri et al. [14] propose optimizations for queries on classification and clustering models in SQL Server. Using model-specific

```
SELECT
 [Forecasting_MIXED].[ModelRegion],
  PredictTimeSeries
   ([Forecasting_MIXED].[Quantity],6),
  PredictTimeSeries
   ([Forecasting_MIXED].[Amount],6)
FROM  [Forecasting_MIXED]
WHERE [ModelRegion] = 'M200 Europe'
  OR  [ModelRegion] = 'M200 Pacific'
```

(a) Forecasting using DMX in SQLServer

```
DEFINE fcst.sales DECIMAL <month>
LIMIT month
TO year 'Yr95' 'Yr96'
FORECAST LENGTH 12
METHOD WINTERS
PERIODICITY 12
ALPHA .5 BETA .5 GAMMA .5
TIME month
FCNAME fcst.sales sales
```

(b) FORECAST command in Oracle

Fig. 4. Comparison of forecast functionalities in SQLServer and Oracle

algorithms, predicates on data mining models are transformed to simple selection predicates, which can then be exploited for access path selection.

Oracle Data Mining (ODM) provides twelve data mining algorithms that address classification, regression, association rules, clustering, attribute importance, and feature selection problems [63]. ODM provides PL/SQL and Java application programming interfaces for model building and model scoring functions as well as a Oracle Data Miner graphical user interface for data analysts who want to use a GUI. Additionally, Oracles offers a `FORECAST` command as part of its OLAP DML (Fig. 4 (b)), which supports linear as well as non-linear regression methods or exponential smoothing [61].

The IBM DB2 Warehouse data mining capabilities provide algorithms for mining tasks such as clustering, classification, association rule mining and regression [7], but no specific time series forecasting methods are supported. Data mining models are represented using the Predictive Model Markup Language (PMML) and stored in relational tables.

Commercial database systems increase the efficiency by pushing statistical computation closer to the database and also offer some advanced time series forecasting methods. However, due to the usage of proprietary languages, forecasting is not integrated within the relational processing and optimization of SQL queries. Additionally, the black box approach makes it difficult to customize, extend and optimize data mining functionality, including the whole forecasting life cycle. Subsequently, models are not handled as first class citizens leading to same drawbacks as discussed for UDFs.

Bi-directional Communication. Finally, a third possibility is to reuse existing statistical tools like R and improve the cooperation between the database and the statistical software system.

Ricardo [18] focuses on large-scale data management systems such as Hadoop and proposes a system where large-scale computations are expressed in JAQL, a high level query interface on top of Hadoop, while R is called for smaller-scale single-node statistical tasks. This requires the programmer to identify scalability of different components of an algorithm, and re-express large-scale matrix operations in terms of JAQL queries.

A second example is the integration of R into the SAP in-memory computing engine. Große et al. [37] developed a shared memory-based data exchange to reduce the communication overhead between R and the database, and, additionally, included R scripts as part of the database execution plan. The latter approach allows multiple R runtimes in parallel processing advanced analytic functionality.

On the commercial side, Oracle R Enterprise [62] embeds the functionality of R inside the Oracle database. A transparency layer supports mapping of R data types to Oracle Database objects and generates SQL transparently from R expressions. Additionally, R scripts can be executed inside the database and the Oracle R Connector for Hadoop enables R users to work with a Hadoop cluster.

Bi-directional communication approaches avoid the re-implementation of statistical functionality and reuse well established statistical software environments, which usually provide advanced time series forecasting functionality. As proposed by Große et al. [37], R scripts can be encapsulated into a native database operator, allowing the processing and optimization of statistical computations within the traditional query execution plan. However, again, the whole statistical computation is treated as black box within the R operator and statistical models are hidden within R scripts. Therefore, models can not be transparently precomputed, materialized and managed within the database system, and optimization possibilities on the forecasting process itself are limited.

4.3 Full Database Integration

In contrast to partial integration approaches, full integration approaches either design a completely new special-purpose database system or extend the core functionality of a traditional database system.

Special-Purpose Database System. A representative of a special-purpose database system is SciDB [11], which targets application domains that involve very large array data such as scientific applications. The SciDB database is organized as collections of n-dimensional arrays and addresses challenges like array storage and partitioning as well as parallel processing of array operations. SciDB supports query patterns such as array slicing and dicing, array scans, and binary

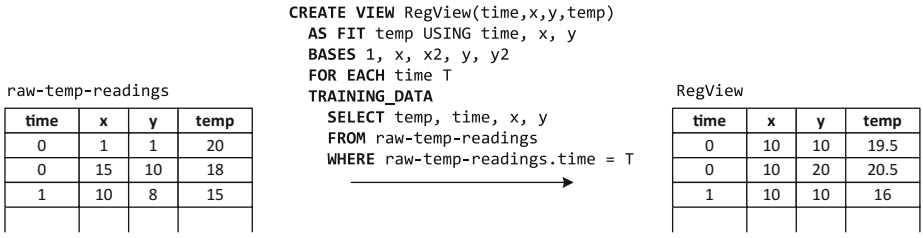

Fig. 5. Model-based views in MauveDB

array operations, but no advanced statistical methods like time series forecasting. The approach of SciDB adapts and optimizes the database system specifically to the target use case and goes far beyond the idea of integrating statistical methods inside a database system. Our goal is to provide time series forecasting within traditional database processing for various use cases and to benefit from existing database technologies, requiring a more general full database integration approach.

Traditional Database System. The MauveDB project [19] integrates statistical modeling inside a DBMS using so-called model-base views. Model-based views generalize the view concept and allow the definition of views as statistical models using extensions to SQL (Fig. 5). Such views can be queried like traditional view leading to new classes of view access operators inside the DBMS. Additionally, MauveDB provides different maintenance strategies that keep models consistent with changes to the data, e.g., no, partial, or full materialization. A similar motivation follows FunctionDB [80], where mathematical functions are treated as first-class citizens inside a DBMS. Queries are answered with discrete points that are computed from piecewise polynomial functions, where the data is discretized as late as possible. These leads to various new relational operations that operate directly on the symbolic representations of the functions.

Akdere et al. [3] expand the idea of MauveDB and propose a Predictive Database Management System (PDBMS), called Longview, that enables declarative predictive queries as well as automatic model training and selection. Longview provides two interfaces for access to its predictive functionality. A direct interface offers direct access to the functionality of the prediction models (regression and classification), whereas a declarative interface is used for high-level access by non-expert users. Prediction models can be built using the `CREATE PREDICTOR` command and then directly queried or referenced in traditional views. Internally, a model manager is responsible for creating materialized models or selecting models in an ad-hoc fashion. However, Akdere et al. [3] provide only a high-level overview over such a system and identify several open research aspects, including automatic selection of materialized models for given cost and accuracy constraints as well as execution and optimization of predictive queries.

Both projects, MauveDB and Longview, integrate statistical methods within a database by viewing models as first class citizens. However, they target statistical methods such as interpolation or classification, and not time series forecasting. Forecasting requires specific time series forecasting methods as well specific model identification, model evaluation, and model maintenance strategies. Furthermore, both approaches require the explicit selection of a model in a query and do not realize declarative and transparent forecast queries.

Besides these general approaches, a number of papers has addressed database aspects of specific mining models. For example, the HAZY project [48] builds upon the work of MauveDB [19] and addresses the incremental maintenance of classification views. The Monte Carlo Database System [46] allows the creation of arbitrary stochastic models for uncertain data and focuses on Monte Carlo

analysis of such models. Simple regression methods have been natively supported by relational database systems for about a decade, and have been incorporated into the SQL language [4]. Other approaches focus on support vector machines [58] and interpolation functions [38].

Additionally, specific time series approaches have been proposed, which we discuss in more detail in the next section.

5 In-DBMS Time Series Forecasting Techniques

We now review dedicated time series forecasting techniques in the database context. Such approaches usually address a specific forecasting method or scenario and discuss individual aspects of the forecasting process in this context.

The processing of declarative forecast queries in traditional databases was first introduced within the Fa System [20]. The main contribution of Duan and Babu [20] is an automatic feature selection approach for forecasting multidimensional time series. A query execution plan in Fa consists of a sequence of transformers, which shift or remove attributes from the input data set; a builder, which computes a forecast model from the transformed data set; and a predictor to make the forecast itself. Fa's plan search is based on an iterative algorithm. Each iteration selects a set of attributes using several heuristics and empirically evaluates five different forecasting methods (regression and machine learning approaches) for the selected attributes, leading to more and more accurate plans over time. Furthermore, an adaptive version of the plan search algorithm for continuous forecast queries is proposed.

Later, Ge and Zdonik [34] proposed an automatic model selection approach for multivariate regression models. Their approach is based on the observation that the best history length of the time series, in terms of accuracy and efficiency, varies according to the requested forecast horizon. An empirical approach iteratively increases the history as well as the number of data points and uses statistical tests of hypotheses to build a single regression model. A skip-list data structure supports the efficient selection of the data at a certain granularity. Additionally, a randomized algorithm is provided that chooses a set of forecast models for a given query workload and maintenance constraints. Maintenance either involves rebuilding the regression model or choosing new properties of the models, i.e., history length and data granularity, where the latter is done after a fixed number of new time series values. Ge and Zdonik also introduce query optimization techniques for range, aggregation, and join queries exploiting the properties of regression models. To compute a join over a future time range, for example, a simple approach would generate all future data points using the regression models and then perform a traditional join on the raw data. In contrast, the second relation could simply use the regression functions of the first relation and solve an equality condition to retrieve the matching tuples. However, such optimization techniques can only be exploited by simple regression function and are not applicable to more sophisticated time series methods, which require additional input data to compute the forecast values (e.g., auto-regressive models).

A formal definition of a forecast operator was developed by Parisi et al. [69] Also, the integration of forecast operators with standard relational operators was explored by identifying simple plan restructuring rules for three relational operators; selection, projection, and union.

In another work, Akdere et al. [2] present optimization techniques for continuous prediction queries using Bayesian Networks as forecasting method. They propose to model point and range-based prediction queries as query plans and introduce different materialization options within a plan. A selection approach finds an execution plan with minimum computation costs for given memory constraints.

The challenge of forecasting high-dimensional data was addressed in the area of online display advertisement [1]. Hundreds of attributes and trillions of attribute combinations have to be forecasted, making it impossible to build a forecast model for each single time series in the database. To solve this issue, only forecast models for a small subset of attribute combinations are built, which are selected manually for seasonality and historical importance. Forecasts for remaining attributes are obtained by exploiting correlations between the attributes. Specifically, three different correlation approaches are evaluated: a Naive Bayes approach that assumes attribute independence, a partwise independence approach that infers combinations of correlated attributes, and a sampling-based approach that computes correlations for a sample of the data.

In the area of data stream management, research has investigated the joint forecasting of multiple data streams. The MUSCLES method [86] uses multivariate linear regression to forecast values of one stream based on the previous values of all streams. MUSCLE is able to adapt to changing correlations among time sequences. SPIRIT [68] finds correlations among data streams by computing the principal components. An auto-regressive model is built directly over the principal components and used for the estimation of missing values.

In terms of model maintenance, Rosenthal and Lehner [73] developed an incremental model adaption approach for simple auto-regressive models and propose a generic approach, called on-demand estimation, for more complex ARIMA models. The parameters of ARIMA models can be estimated using the maximum likelihood approach, which tries to maximizes the probability of reproducing the training data from the given parameters. On-demand estimation incrementally maintains the so called likelihood function and triggers model adaption if new time series values lead to significant changes in the function's optimum.

Maintenance issues have also been discussed in the context of streaming databases and sensor networks. Tulone and Madden [81] propose an error-based model evaluation strategy for auto-regressive models. Model adaption is triggered based on two thresholds, which distinguish outlier values and distribution changes in the data. The latter suggest that re-learning the model might be necessary. In contrast, the sensor data management architecture PRESTO [52] retrains models periodically. Ikonomovska et al. [44] propose an incremental

stream mining algorithm for regression and model trees, including drift detection and model adaptation to maintain accurate and updated regression models at any time.

To sum up, the need for integrating analytical methods into traditional databases has been identified by many existing research projects, addressing general approaches as well as specific forecasting methods. However, non of the existing approaches provide a complete solution for in-DBMS forecasting, including declarative forecast queries, arbitrary forecasting methods, relational query processing, query optimization, forecast model maintenance, transparent model reuse, and automatic model selection. Following the discussion of the general forecasting process from Sect. 3.1, we now introduce an architecture that integrates the whole forecasting life cycle natively into an existing DBMS and, additionally, benefits from existing work on in-DBMS forecasting techniques.

6 A Flash-Forward Database System

In contrast to flash back queries that allow a view on the data in the past, we developed a *Flash-Foreward Database System*. We explain the necessary extensions to a traditional DBMS from two angles. First, in Subsect. 6.1, we investigate changes to the different types of schemas of a DBMS, which are usually described by the ANSI/SPARC architecture. Subsequently, in Subsect. 6.2, we discuss the actual implementation of a forecast-enabled database management system.

6.1 ANSI/SPARC Architecture

The ANSI/SPARC architecture forms an abstract design standard for a database management systems and gives a general architecture for database functions, interfaces, and usages. The objective of the three-level architecture is to separate the users' view of the data from the way that it is physically represented. Specifically, the use of the data is described in the external schema, the meaning of the data in the conceptual schema, and the data storage in the internal schema. Time series forecasting consists of two major data entities — time series and forecast models — that have to be arranged within the ANSI-SPARC architecture. We now systematically study each of the three levels of the architecture and discuss where we have to add new concepts and where we can reuse existing concepts from the ANSI-SPARC architecture (see Fig. 6) [25].

External Schema. The external schema in the traditional ANSI/SPARC architecture consists of user-defined data views, which can be seen as virtual tables storing the results of specific queries. Time series can just be seen as a special view that ensure the representation of the data as time series. A *time series view* requires at least a time attribute containing discrete points in time and another attribute exhibiting the measurements at these specified moments, for example:

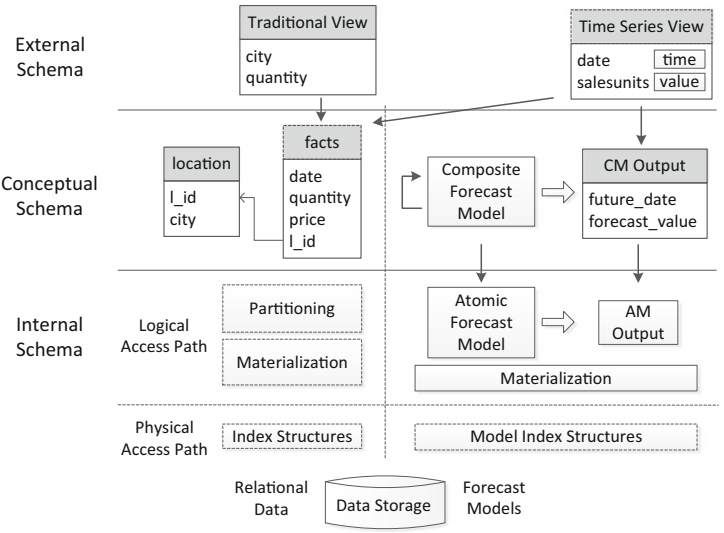

Fig. 6. Integration of forecasting within the ANSI/SPARC architecture

```
CREATE VIEW timeSeriesView AS
  SELECT    MONTH(date) AS month, SUM(salesunits) as sales
  FROM      facts f, products p
  WHERE         f.product_id = p.product_id
  AND           p.pname = 'audio'
  AND             month in (now() - 3 months, now() + 3 months)
  GROUP BY  month
```

A time series view can represent historical values, forecast values, or both. If forecast values are involved, the time series view has to be defined by a query requesting future values. It might contain further information such as standard deviation or prediction intervals, which clearly distinguish future from historical values. Once real values are available, they replace the forecast values.

Conceptual Schema. The conceptual level includes a data schema that describes available entities, their relationship and contained attributes and can be seen as an abstraction from the internal data representation. Likewise, a *composite forecast model* is defined as a conceptual abstraction from a concrete *atomic forecast model*. A composite model might directly refer to a single atomic forecast model from the internal schema, representing a simple forecast of a time series, e.g., sales units of audio devices. However, composite forecast models can also describe a (hierarchical) forecast model composition. When forecasting sales units of audio devices in Germany, for example, the forecasting can be decomposed into forecasts of the sales units for all German states, or further down in the hierarchy, sales units of all German cities. The composite forecast model can define a hierarchical forecast composition referring further composite models on multiple hierarchy levels and, on the leaf level, ultimately refer to atomic forecast

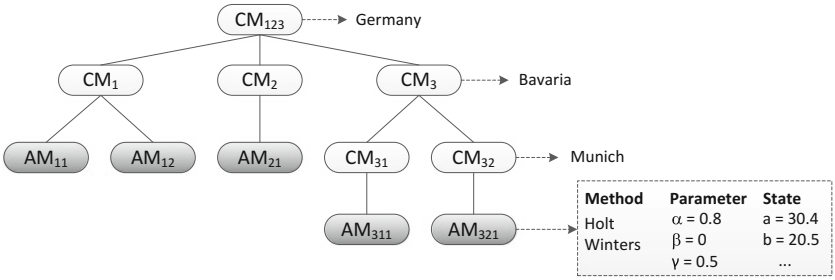

Fig. 7. Example model composition

models defined in the internal schema (see Example in Fig. 7). Multiple atomic forecast models, with different forecasting methods or parameter combinations, might be referenced by one composite model on the leaf level, enabling ensemble forecasting.

With respect to the external layer, each composite forecast model is linked with a single time series view from the external schema. It further defines a single output, the *composite model output* (CM output), which is a special table complying to the same rules as the time series view and exhibits the same hierarchical forecast model composition as defined for the associated composite forecast model. The forecast values, i.e., the composite forecast model output, are computed by a weighted linear combination of the referenced forecast models according to the defined forecast model composition.

Internal Schema. The logical and the physical data access paths are defined in the internal schema of the ANSI-SPARC architecture. Logical access paths refer to data organization aspects like partitioning and materialization, whereas the physical access paths define low level access structures like indexes. Likewise, atomic forecast models are defined that represent a non-decomposable forecast model. A single atomic forecast model is represented by input and output definitions, the forecasting method, model parameters, and the current model state (see Subsect. 3.1). Here, the input is the data as defined in the associated time series view, referenced through the connected composite forecast model. The forecasting method is chosen from a *forecast model catalog* that represents all forecasting methods available in the DBMS and is predefined with respect to the application domain (similar to the approach of Longview [3]). The output of atomic forecast models is represented by a special data structure called *atomic forecast model output* (AM Output). Optionally, additional attributes might be included in the model output (e.g., prediction intervals).

Traditionally, materialization is performed to precompute complex database queries. Similarly, composite and atomic models can be materialized for faster query response times. Materialized models might store composition rules, model parameters, model states, or even the model output, i.e., forecast results. On the physical access paths, specific model specific index structures might be

applied [27,34]. Additionally, traditional or customized time series index structures ensure efficient processing of time series data and access of time series values in a subsequent order.

6.2 DBMS Architecture

In contrast to the ANSI/SPARC architecture, which mainly describes interfaces, we now discuss the actual realization of a flash-forward database system [28]. Figure 8 shows the main components of a database management systems, exemplary on the open-source DBMS PostgreSQL [70]. Our extensions to traditional database components are shown by grey boxes. In what follows, we shortly outline the core idea of the main components, more details can be found in [27–30].

First of all, declarative forecast queries require the extension of the parser so that forecast-specific keywords (e.g., forecast horizon, forecasting attributes, forecasting method) are recognized. After parsing, the statement is identified as complex (e.g., select, insert, delete) or simple (e.g., create table) by the traffic cop. Simple utility commands are processed by a dedicated component, which contains forecast model specific utility commands (e.g., create model, drop model). This enables a database administrator to explicitly create and delete models. Complex statements are planned by the optimizer. Hereby, the existing optimizer is extended with (1) new forecast-specific physical operators, (2) new cost models for those operators as well as (3) new optimizer rules.

Following the general forecasting process, forecast operators decouple model creation and model usage functionality. The `CreateModel` operator is responsible for model creation. It receives a time series view as input and outputs a set of forecast models. Subsequently, the `Forecast` operator receives as input a set of forecast models and outputs a time series relation containing corresponding forecast values. In case of ad hoc forecast queries, these two operators appear jointly, with the `Forecast` operator sitting on top of the `CreateModel` operator. However, the separation of both operators enables the transparent reuse of materialized models.

Materialized models are handled by two new components — *model matching* and *model maintenance* [27]. Model matching is responsible for finding suitable models for a given forecast query, including atomic forecast models and potential model compositions. Depending on the query type, model matching can be accessed by either the optimizer or executor. In contrast, model maintenance is performed after insert statements. It is responsible for finding models that are based on those inserts. Model maintenance includes a model update step, an evaluation step and, optionally, a parameter re-estimation step.

Besides, the transparent reuse of materialized models, the optimizer is also responsible for processing ad hoc forecast queries that require the creation of a model at query runtime. Hereby, we exploit traditional database sampling techniques to reduce the amount of processed data by the `CreateModel` operator [29]. For example, one optimization techniques reduces the time series length for parameter estimation.

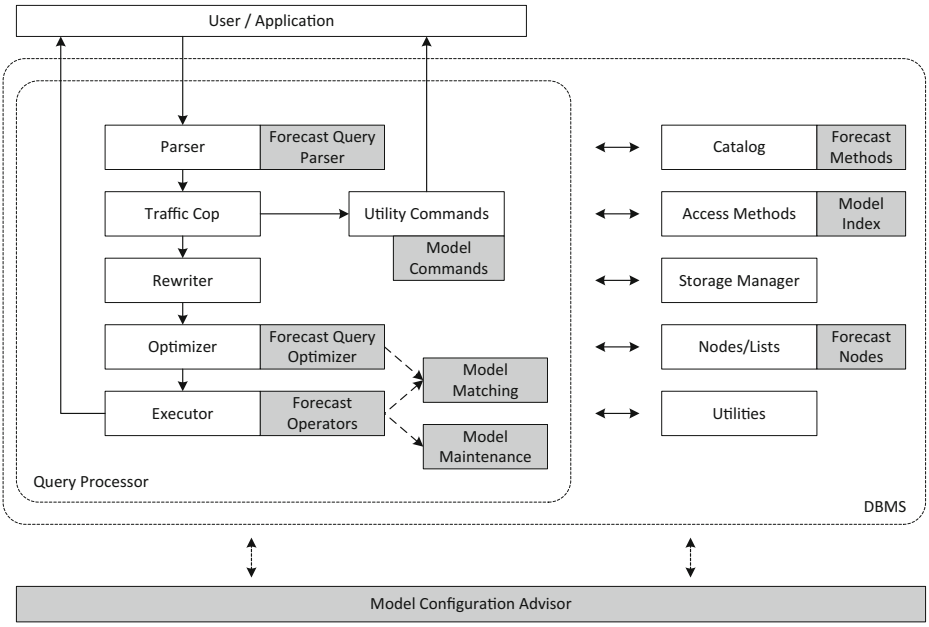

Fig. 8. Architecture of a flash-forward database system

The discussed query processing components are supported by other modules. The catalog stores meta data about atomic and composite forecast models as well as the previously mentioned forecasting method catalog. As discussed in Subsect. 6.1 access methods are extended with model-specific access structures, supporting the model matching and model maintenance components [27]. Information about internal query structures and query plans are stored in nodes and lists in PostgreSQL. Thus, new nodes for forecast queries have to be added. The remaining two components, the storage manager and utilities, containing support functions, are currently not touched by our extensions. Traditional tables are used to store time series data as well as models, which enables the direct reuse of the different functionalities of a storage manager.

Besides the extension of internal components, one additional external component, the *model configuration advisor*, is available [30]. Similar to a traditional index or materialized view advisor, which proposes a physical design of indexes and materialized views to the database system, the model configuration advisor recommends a physical design of forecast models. In contrast to traditional advisers, which usually focus on minimizing the runtime of a given workload (optionally giving some storage constraints), the optimization objective of the model configuration advisor is twofold — minimize the query runtime and maximize the forecast accuracy. The technical challenge of the advisor comes from the fact that there are no known ways to estimate the accuracy of a physical design of forecast models without actually deploying and querying it. Hence, the

model advisor is based on an iterative process that, based on heuristics, selects a set of candidate time series in each iteration for which a model should be built and analyzed. Parameters of the advisor like the number of candidate models are automatically tuned in a control phase.

7 Conclusions and Future Work

The need for integrating statistical methods into databases has been identified by many existing research projects, addressing general approaches as well as specific statistical methods. In this article, we provided a review of existing work and discussed its applicability for supporting a transparent model-based database system, where we specifically focused on forecast models. Based on the traditional ANSI/SPARC architecture, we introduced a novel forecast-enabled database management system, the flash-forward database system. Our approach belongs to the class of full database integration approaches and integrates the whole forecasting life cycle.

Recall the application scenarios and associated requirements presented in Sect. 2. The proposed flash-forward database systems enables forecasting by non-expert users (e.g., supply chain managers) and hides the complexity of the forecasting process. It allows the integration of a suite of forecasting techniques and domain-specific forecasting methods (e.g., for energy demand and supply forecasting). The reuse of materialized forecast models enables the processing of forecast queries in real-time as required, for example, in energy load balancing or display advertisement. The maintenance component continuously and efficiently adapts models to changes in the time series behavior (e.g., weather changes in the energy domain). Finally, the model configuration advisor selects a physical design of forecast models for large multi-dimensional data sets (e.g., in producting planning), balancing query efficiency and forecast accuracy.

Although we specifically focused on forecast models in this article, many of the discussed challenges, foundations, and concepts can be applied to other statistical models. We have taken a first step towards a model-based database system and opened up interesting opportunities for further research. We conclude by mentioning a few of them:

- *Configuration Maintenance:* How can we maintain a configuration of forecast models in an online fashion? Can we develop incremental algorithms that avoid the complete re-execution of the forecast model advisor?
- *Parallelized Query Execution:* How can we improve the execution of adhoc forecast queries that require the creation of a new model? Can we develop efficient operators that exploit parallelization opportunities of modern (heterogeneous) hardware environments?
- *Lineage:* Can we provide the user with information about the origin as well as reliability of the query result?

References

1. Agarwal, D., Chen, D., Lin, L., Shanmugasundaram, J., Vee, E.: Forecasting high-dimensional data. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 1003–1012 (2010)
2. Akdere, M., Çetintemel, U., Upfal, E.: Database-support for continuous prediction queries over streaming data. Proc. VLDB Endowment **3**, 1291–1301 (2010)
3. Akdere, M., Cetintemel, U., Riondato, M., Upfal, E., Zdonik, S.B.: The case for predictive database systems: opportunities and challenges. In: Fifth Biennial Conference on Innovative Data Systems Research, pp. 167–174 (2011)
4. Alur, N., Haas, P., Momiroska, D., Read, P., Summers, N., Totanes, V., Zuzarte, C.: DB2 UDB's High Function Business Intelligence in e-Business. IBM Redbook Series (2002)
5. Andersen, T.G., Bollerslev, T., Lange, S.: Forecasting financial market volatility: sample frequency vis-a-vis forecast horizon. J. Empirical Finan. **6**, 457–477 (1999)
6. Apache. Apache Mahout (2013). http://mahout.apache.org/
7. Ballard, C., Rollins, J., Ramos, J., Perkins, A., Hale, R., Doerneich, A., Milner, E.C., Chodagam, J.: Dynamic Warehousing: Data Mining Made Easy. IBM Redbooks Series (2007). http://www.redbooks.ibm.com/redbooks/pdfs/sg247418.pdf
8. Bontempi, G., Ben Taieb, S., Le Borgne, Y.-A.: Machine learning strategies for time series forecasting. In: Aufaure, M.-A., Zimányi, E. (eds.) eBISS 2012. LNBIP, vol. 138, pp. 62–77. Springer, Heidelberg (2013)
9. Box, G.E.P., Jenkins, G.M., Reinsel, G.C.: Time Series Analysis: Forecasting and Control, 4th edn. Wiley, New York (2008)
10. Brockwell, P.J., Davis, R.A.: Introduction to Time Series and Forecasting. Prentice Hall, Englewood Clifs (2002)
11. Brown, P.G.: Overview of sciDB: large scale array storage, processing and analysis. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 963–968 (2010)
12. Cetintas, S., Chen, D., Si, L., Shen, B., Datbayev, Z.: Forecasting counts of user visits for online display advertising with probabilistic latent class models. In: International Conference on Research and Development in Information Retrieval, pp. 1217–1218 (2011)
13. Chatfield, C.: Time-Series Forecasting. Chapman & Hall, Boca Raton (2000)
14. Chaudhuri, S., Narasayya, V., Sarawagi, S.: Efficient evaluation of queries with mining predicates. In: Proceedings of the 18th International Conference on Data Engineering, pp. 529–540 (2002)
15. Cohen, J., Dolan, B., Dunlap, M., Hellerstein, J.M., Welton, C.: MAD skills: new analysis practices for big data. Proc. VLDB Endowment **2**, 1481–1492 (2009)
16. Dannecker, L., Böhm, M., Lehner, W., Hackenbroich, G.: Forcasting evolving time series of energy demand and supply. In: Eder, J., Bielikova, M., Tjoa, A.M. (eds.) ADBIS 2011. LNCS, vol. 6909, pp. 302–315. Springer, Heidelberg (2011)
17. Dannecker, L., Schulze, R., Böhm, M., Lehner, W., Hackenbroich, G.: Context-aware parameter estimation for forecast models in the energy domain. In: Bayard Cushing, J., French, J., Bowers, S. (eds.) SSDBM 2011. LNCS, vol. 6809, pp. 491–508. Springer, Heidelberg (2011)
18. Das, S., Sismanis, Y., Beyer, K.S., Gemulla, R., Haas, P.J., McPherson, J.: Ricardo: integrating r and hadoop. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 987–998 (2010)

19. Deshpande, A., Madden, S.: MauveDB: supporting model-based user views in database systems. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, pp. 73–84 (2006)
20. Duan, S., Babu, S.: Processing forecasting queries. In: Proceedings of the VLDB Endowment, pp. 711–722 (2007)
21. European Commission. Energy Roadmap 2050. Brussels (2011)
22. Fang, L., LeFevre, K.: Splash: ad-hoc querying of data and statistical models. In: Proceedings of the 13th International Conference on Extending Database Technology, pp. 275–286 (2010)
23. Feng, H.: Performance problems of forecasting systems. In: 15th East-European Conference on Advances in Databases and Information Systems, pp. 254–261 (2011)
24. Feng, X., Kumar, A., Recht, B., Ré, C.: Towards a unified architecture for in-rdbms analytics. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, pp. 325–336 (2012)
25. Fischer, U., Dannecker, L., Siksnys, L., Rosenthal, F., Boehm, M., Lehner, W.: Towards integrated data analytics: time series forecasting in dbms. Datenbank-Spektrum, 1–9 (2012)
26. Fischer, U., Kaulakienė, D., Khalefa, M.E., Lehner, W., Pedersen, T.B., Šikšnys, L., Thomsen, C.: Real-time business intelligence in the MIRABEL smart grid system. In: Castellanos, M., Dayal, U., Rundensteiner, E.A. (eds.) BIRTE 2012. LNBIP, vol. 154, pp. 1–22. Springer, Heidelberg (2013)
27. Fischer, U., Rosenthal, F., Böhm, M., Lehner, W.: Indexing forecast models for matching and maintenance. In: IDEAS, pp. 26–31 (2010)
28. Fischer, U., Rosenthal, F., Lehner, W.: F2DB: the flash-forward database system. In: Proceedings of the 28th International Conference on Data Engineering, pp. 1245–1248 (2012)
29. Fischer, U., Rosenthal, F., Lehner, W.: Sample-based forecasting exploiting hierarchical time series. In: Proceedings of the 16th International Database Engineering and Applications Sysmposium, pp. 120–129 (2012)
30. Fischer, U., Schildt, C., Hartmann, C., Lehner, W.: Forecasting the data cube: a model configuration advisor for multi-dimensional data sets. In: Proceedings of the 29th International Conference on Data Engineering (2013)
31. Fliedner, G.: Hierarichal forecasting issues and use guidelines. Ind. Manage. Data Syst. **101**, 5–12 (2001)
32. Ganapathi, A., Kuno, H., Dayal, U., Wiener, J.L., Fox, A., Jordan, M., Patterson, D.: Predicting multiple metrics for queries: better decisions enabled by machine learning. In: Proceedings of the 25th International Conference on Data Engineering, pp. 592–603 (2009)
33. Gardner Jr, E.S.: Exponential smoothing: the state of the art. Int. J. Forecast. **4**, 1–28 (1985)
34. Ge, T., Zdonik, S.B.: A skip-list approach for efficiently processing forecasting queries. Proc. VLDB Endowment **1**, 984–995 (2008)
35. Ghoting, A., Krishnamurthy, R., Pednault, E., Reinwald, B., Sindhwani, V., Tatikonda, S., Tian, Y., Vaithyanathan, S.: SystemML: declarative machine learning on mapreduce. In: Proceedings of the 2011 IEEE 27th International Conference on Data Engineering, pp. 231–242 (2011)
36. Gooijera, J.G.D., Hyndman, R.J.: 25 years of time series forecasting. Int. J. Forecast. **22**, 443–473 (2006)

37. Große, P., Lehner, W., Weichert, T., Färber, F., Li, W.-S.: Bridging two worlds with rice integrating r into the sap in-memory computing engine. Proc. VLDB Endowment **4**, 1307–1317 (2011)
38. Grumbach, S., Rigaux, P., Segoufin, L.: Manipulating interpolated data is easier than you thought. In: Proceedings of the 26th International Conference on Very Large Data Bases, pp. 156–165 (2000)
39. Harries, M., Horn, K.: Detecting concept drift in financial time series prediction using symbolic machine learning. In: Proceedings of the 8th Australian Joint Conference on Artificial Intelligence, pp. 91–98 (1995)
40. Holt, C.C.: Forecasting seasonals and trends by exponentially weighted moving averages. Int. J. Forecast. **20**, 5–10 (2004)
41. Hyndman, R.J., Khandakar, Y.: Automatic time series forecasting: the forecast package for R. J. Stat. Softw. **27**, 1–22 (2008)
42. Hyndman, R.J., Koehler, A.B., Snyder, R.D., Grose, S.: A state space framework for automatic forecasting using exponential smoothing methods. Int. J. Forecast. **18**, 439–454 (2002)
43. Hyndman, R.J., Kostenko, A.V.: Minimum sample size requirements for seasonal forecasting models. Foresight: the Int. J. Appl Forecast. **6**, 12–15 (2007)
44. Ikonomovska, E., Gama, J., Džeroski, S.: Learning model trees from evolving data streams. Data Min. Knowl. Discov. **23**, 128–168 (2011)
45. Imieliński, T., Virmani, A.: Msql: a query language for database mining. Data Min. Knowl. Discov. **3**, 373–408 (1999)
46. Jampani, R., Xu, F., Wu, M., Perez, L.L., Jermaine, C., Haas, P.J.: Mcdb: a monte carlo approach to managing uncertain data. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 687–700 (2008)
47. Kimball, R., Ross, M.: The Data Warehouse Toolkit. Wiley, New York (2002)
48. Koc, M.L., Ré, C.: Incrementally maintaining classification using an rdbms. Proc. VLDB Endowment **4**, 302–313 (2011)
49. Kraska, T., Talwalkar, A., Duchi, J., Griffith, R., Franklin, M.J., Jordan, M.: Mlbase:a distributed machine learning system. In: 6th Biennial Conference on Innovative Data Systems Research (2013)
50. Kusters, U., McCullough, B., Bell, M.: Forecasting software: past, present and future. Int. J. Forecast. **22**, 599–615 (2006)
51. Lazarescu, M.M., Venkatesh, S., Bui, H.H.: Using multiple windows to track concept drift. Intell. Data Anal. J., 1–28 (2003)
52. Li, M., Ganesan, D., Shenoy, P.: Presto: feedback-driven data management in sensor networks. In: Proceedings of the 3rd Conference on Networked Systems Design & Implementation, pp. 23–23 (2006)
53. Makridakis, S.: Accuracy measures: theoretical and practical concerns. Int. J. Forecast. **9**, 527–529 (1993)
54. Makridakis, S., Hibon, M.: The M3-Competition: results, conclusions and implications. Int. J. Forecast. **16**, 451–476 (2000)
55. Matlab. The language of technical computing (2012). http://www.mathworks.com/products/matlab/
56. Meek, C., Chickering, D.M., Heckerman, D.: Autoregressive tree models for time-series analysis. In: SIAM International Conference on Data Mining (2002)
57. Mentzer, J.T., Bienstock, C.C.: The seven principles of sales-forecasting systems. Supply Chain, Manage. Rev. **11**, 76–83 (1998)

58. Milenova, B.L., Yarmus, J.S., Campos, M.M.: Svm in oracle database 10g: removing the barriers to widespread adoption of support vector machines. In: Proceedings of the VLDB Endowment, pp. 1152–1163 (2005)
59. Mills, T.C.: Time Series Techniques for Economists. Business & Economics (1991)
60. Müller, K.-R., Smola, A.J., Rätsch, G., Schölkopf, B., Kohlmorgen, J., Vapnik, V.: Predicting time series with support vector machines. In: Gerstner, W., Hasler, M., Germond, A., Nicoud, J.-D. (eds.) ICANN 1997. LNCS, vol. 1327, pp. 999–1004. Springer, Heidelberg (1997)
61. Oracle OLAP DML Reference 11g. Forecast - dml statement (2012). http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_commands_1052.htm
62. Oracle R. Enterprise user's guide (2012). http://docs.oracle.com/cd/E27988_01/doc/doc.112/e26499.pdf
63. Oracle White Paper. Oracle data mining 11g release 2 - competing on in-database analytics (2012)
64. Ordonez, C.: Programming the k-means clustering algorithm in sql. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 823–828 (2004)
65. Ordonez, C., Pitchaimalai, S.K.: Bayesian classifiers programmed in sql. IEEE Trans. Knowl. Data Eng. **22**, 139–144 (2010)
66. Ordonez, C., Pitchaimalai, S.K.: One-pass data mining algorithms in a dbms with udfs. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, pp. 1217–1220 (2011)
67. Panda, B., Herbach, J.S., Basu, S., Bayardo, R.J.: Planet: massively parallel learning of tree ensembles with mapreduce. Proc. VLDB Endowment **2**, 1426–1437 (2009)
68. Papadimitriou, S., Sun, J., Faloutsos, C.: Streaming pattern discovery in multiple time-series. In: Proceedings of the 31st International Conference on Very Large Data Bases, pp. 697–708 (2005)
69. Parisi, F., Sliva, A., Subrahmanian, V.S.: Embedding forecast operators in databases. In: Benferhat, S., Grant, J. (eds.) SUM 2011. LNCS, vol. 6929, pp. 373–386. Springer, Heidelberg (2011)
70. PostgreSQL (2012). http://www.postgresql.org/
71. R Development Core Team. R: A language and environment for statistical computing, reference index version 2.1.1. R Foundation for Statistical Computing (2012). http://www.r-project.org
72. Ramanathan, R., Engle, R., Granger, C.W.J., Vahid-Araghi, F., Brace, C.: Short-run forecasts of electricity loads and peaks. Int. J. Forecast. **13**(2), 161–174 (1997)
73. Rosenthal, F., Lehner, W.: Efficient in-database maintenance of ARIMA models. In: Bayard Cushing, J., French, J., Bowers, S. (eds.) SSDBM 2011. LNCS, vol. 6809, pp. 537–545. Springer, Heidelberg (2011)
74. Sadri, R., Zaniolo, C., Zarkesh, A.M., Adibi, J.: A sequential pattern query language for supporting instant data mining for e-services. In: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 653–656 (2001)
75. SAS. Business intelligence software (2012). http://www.sas.com
76. Schmidberger, M., Morgan, M., Eddelbuettel, D., Yu, H., Tierney, L., Mansmann, U.: State-of-the-art in parallel computing with R. J. Stat. Softw. **31**, 1–27 (2009)
77. Shalev-Shwartz, S., Srebro, N.: SVM optimization: inverse dependence on training set size. In: Proceedings of the 25th International Conference on Machine Learning, pp. 928–935 (2008)
78. SPSS. IBM SPSS Statistics (2012). http://www-01.ibm.com/software/analytics/spss/

79. SQL Server. Data Mining Algorithms - Books Online for SQL Server 2012 (2012). http://msdn.microsoft.com/en-us/library/ms175595.aspx
80. Thiagarajan, A., Madden, S.: Querying continuous functions in a database system. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 791–804 (2008)
81. Tulone, D., Madden, S.: PAQ: time series forecasting for approximate query answering in sensor networks. In: Römer, K., Karl, H., Mattern, F. (eds.) EWSN 2006. LNCS, vol. 3868, pp. 21–37. Springer, Heidelberg (2006)
82. Turner, J.: The planning of guaranteed targeted display advertising. Oper. Res. **60**, 18–33 (2012)
83. Wagner, N., Michalewicz, Z., Khouja, M., McGregor, R.: Time series forecasting for dynamic environments: the dyfor genetic program model. IEEE Trans. Evol. Comput. **11**, 433–452 (2007)
84. Wang, H., Zaniolo, C., Luo, C.R.: ATLAS: a small but complete sql extension for data mining and data streams. In: Proceedings of the 29th International Conference on Very Large Data Bases, pp. 1113–1116 (2003)
85. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Mach. Learn. **23**, 69–101 (1996)
86. Yi, B., Sidiropoulos, N.D., Johnson, T., Jagadish, H.V., Faloutsos, C., Biliris, A.: Online data mining for co-evolving time sequences. In: Proceedings of the 16th International Conference on Data Engineering, pp. 13–22 (2000)
87. Zhang, C., Sun, S., Yu, G.: A bayesian network approach to time series forecasting of short-term traffic flows. In: Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems, pp. 216–221 (2004)
88. Zhang, G., Eddy-Patuwo, B., Hu, M.Y.: Forecasting with artificial neural networks: the state of the art. Int. J. Forecast. **14**, 35–62 (1998)
89. Zhang, Y., Zhang, W., Yang, J.: I/O-efficient statistical computing with RIOT. In: Proceedings of the 26th International Conference on Data Engineering, pp. 1157–1160 (2010)
90. Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: Proceedings of the 20th International Conference on Machine Learning, pp. 928–936 (2003)