Dieses Dokument ist eine Zweitveröffentlichung (Postprint) /

This is a self-archiving document (accepted version):

Ulrike Fischer, Lars Dannecker, Laurynas Siksnys, Frank Rosenthal, Matthias Boehm, Wolfgang Lehner

Towards Integrated Data Analytics: Time Series Forecasting in DBMS

Erstveröffentlichung in / First published in:

Datenbank-Spektrum. 2013, 13 (1), S. 45–53 [Zugriff am: 27.01.2023]. Springer. ISSN 1610-1995.

DOI: http://dx.doi.org/10.1007/s13222-012-0108-4

Diese Version ist verfügbar / This version is available on:

https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-831192

SLUB
Wir führen Wissen.

TECHNISCHE UNIVERSITÄT DRESDEN

QUCOSA
Quality Content of Saxony

Final edited form was published in "Datenbank-Spektrum" 13 (1), S. 45–53. ISSN: 1610-1995
http://dx.doi.org/10.1007/s13222-012-0108-4

Towards Integrated Data Analytics: Time Series Forecasting in DBMS

**Ulrike Fischer · Lars Dannecker · Laurynas Siksnys ·
Frank Rosenthal · Matthias Boehm · Wolfgang Lehner**

Abstract Integrating sophisticated statistical methods into database management systems is gaining more and more attention in research and industry in order to be able to cope with increasing data volume and increasing complexity of the analytical algorithms. One important statistical method is time series forecasting, which is crucial for decision making processes in many domains. The deep integration of time series forecasting offers additional advanced functionalities within a DBMS. More importantly, however, it allows for optimizations that improve the efficiency, consistency, and transparency of the overall forecasting process. To enable efficient integrated forecasting, we propose to enhance the traditional 3-layer ANSI/SPARC architecture of a DBMS with forecasting functionalities. This article gives a general overview of our proposed enhancements and presents how forecast queries can be processed using an example from the energy data management domain. We conclude with open research topics and challenges that arise in this area.

Keywords Time series forecasting · DBMS · Architecture · Challenges

Matthias Boehm is currently visiting IBM Almaden Research Center, San Jose, CA, USA.

U. Fischer (🖂) · F. Rosenthal · M. Boehm · W. Lehner
Technische Universität Dresden, Nöthnitzer Str. 46,
01187 Dresden, Germany
e-mail: ulrike.fischer@tu-dresden.de

F. Rosenthal
e-mail: frank rosenthal@tu-dresden.de

M. Boehm
e-mail: matthias.boehm@tu-dresden.de

W. Lehner
e-mail: wolfgang.lehner@tu-dresden.de

L. Dannecker
SAP Research Dresden, Chemnitzer Str. 48, 01187 Dresden, Germany
e-mail: lars.dannecker@sap.com

L. Siksnys
Center for Data-intensive Systems, Aalborg University,
Selma Lagerlöfs Vej 300, 9220 Aalborg, Denmark
e-mail: siksnys@cs.aau.dk

1 Introduction

Modern data analysis in database management systems involves increasingly sophisticated statistical methods that go well beyond the rollup and drilldown over simple aggregates of traditional BI. Additionally, a rising need for ad-hoc data mining can be observed, which requires fast and automatic processing of complex statistical queries [4, 33]. There exists a wide variety of external statistic tools that provide rich statistical functionality but often suffer from limited scalability and efficiency [8]. As most data subject to analysis is stored in database management systems, which provide powerful mechanism for accessing, partitioning, filtering, and indexing data, a rising trend addresses the integration of statistical methods inside a DBMS. This is especially valid for upcoming in-memory databases [12] that allow fast data analysis per se.

Many interesting integration aspects concerning specific statistical methods (e.g., classification [23], regression [9]) as well as more general approaches [8, 18] have been discussed in the past. In this article, we explicitly focus on integration aspects and challenges of *time series forecasting*, which is an important statistical method and crucial for manual and automatic decision making processes in many domains [17].

For example, the following three use cases require very accurate and efficient time series forecasting:

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum" 13 (1), S. 45–53. ISSN: 1610-1995
http://dx.doi.org/10.1007/s13222-012-0108-4

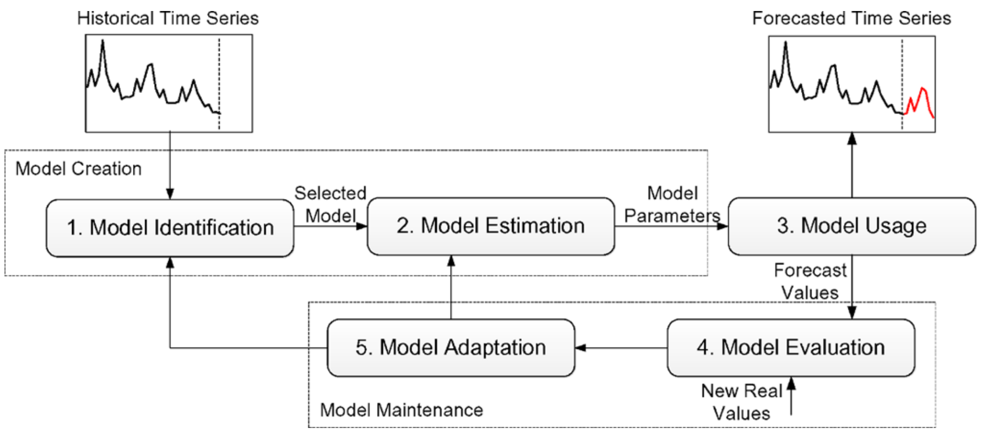


Fig. 1 General Forecasting Process

Forecasting Sales Forecasting forms the basis for planning in many commercial decision-making-processes (e.g., inventory or production planning). Typically, time series data is stored in database management systems and complex analytical (often reoccurring) decision queries including forecast queries are run on this data. Typical forecast queries involve many different time series according to different dimensions (e.g., product, brand, store) and are often submitted by decision managers that do not know much about statistical forecast models. This results in the fact that forecasting should be as transparent to the user as possible.

Forecasting Energy Data Renewable Energy Sources (e.g., windmills) pose the challenge that production is dependent on external factors and thus cannot be planned like traditional energy sources. In this context, accurate forecasting is crucial for balancing energy supply and demand to reduce the overall energy mismatch and penalties paid for any kind of imbalances [3]. In addition, the need for fast response times to react to new market situations and a continuous stream of new production and consumption measurements requires very efficient forecasting functionalities.

Traffic In the automotive domain, predictive location-based services deliver relevant content information to drivers according to routes they undertake (e.g., information about gas stations and traffic conditions) [22]. Such services continuously collect millions of user locations, store them in a moving object database, and run expensive prediction algorithms to determine movement trajectories of all users. Prediction algorithms also involve time series forecasting [34] (e.g., speed prediction) to allow timely delivery of high quality content to users. It is possible only if accurate and up-to-date forecasts are available at any time.

All three use cases are based on the traditional model-based time series forecasting process (Fig. 1) that consists of three main phases—*model creation* (identification and estimation), *model usage* (forecasting), and *model maintenance* (evaluation and adaption). First, the model creation phase involves selecting and building a stochastic model that captures the dependency of future on past data. This is an expensive process as parameter estimation of many sophisticated models involves numerical optimization methods that iterate several times over the base data. However, once a model is created and parameters are estimated, it can cheaply be used over and over again to forecast future values of the time series (model usage). The model maintenance phase evaluates new actual data of the time series and triggers possible adaption of the forecast model. This is computationally expensive as well, as most parameters cannot be maintained incrementally and thus, again a parameter estimation is required.

A tight coupling of the described forecasting process within a DBMS (1) ensures consistency between data and models, (2) increases efficiency by reducing data transfer and exploiting database related optimization techniques, and (3) enables declarative forecast queries for any user. In the context of integrating statistical methods into databases, we can observe two main research directions. Partial integration approaches do not change the database itself by reusing existing statistical tools like R and improving their cooperation with the DBMS [8, 18]. For example, Große et. al. [18] developed a shared memory-based data exchange to reduce the communication overhead between R and the database. While such approaches are more general and easier to realize, they do not allow for optimizations on the forecasting process itself. In contrast, we argue for a full integration approach that extends all layers of a traditional DBMS leading to a higher initial effort but offering much higher optimization potential.

In the remainder of this paper, we first summarize the requirements of the presented use cases and shortly discuss related work for each of these requirements (Sect. 2). We then present a general forecasting architecture that addresses the defined requirements and explain forecast query processing in the use case of forecasting energy data (Sect. 3). Finally, we give a brief overview over research topics and challenges in Sect. 4 and conclude in Sect. 5.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum" 13 (1), S. 45–53. ISSN: 1610-1995
http://dx.doi.org/10.1007/s13222-012-0108-4

2 Requirements and Related Work

Following the presented use cases, we can derive general requirements of forecasting within a database:

Advanced Forecasting Functionality First of all, the database system should provide advanced statistical forecasting functionalities that provide high accuracy for various use cases and time series data. For example in the energy domain multi-equation forecast models are often necessary to achieve reasonable accuracy [27]. Also, new forecasting methods required by applications should be easily addable. Major commercial DBMSs support only a limited amount of such forecasting functionality. For example, Oracle offers linear as well as non-linear regression methods or exponential smoothing as part of its OLAP DML [25]. The Data Mining Extension (DMX) in Microsoft SQL Server supports a hybrid forecast method consisting of ARIMA and autoregressive trees [1].

Declarative Forecast Queries Forecast queries should follow the traditional SQL interface and offer a simple language extension usable for any user. For example, Duan et. al. [10] proposed a simple `FORECAST` keyword to specify declarative forecast queries.

Integration into Relational Query Processing Forecast queries should be seamlessly integrated into standard relational query processing allowing arbitrary forecast queries as well queries on forecasted query results (e.g., joins of forecasted and historical data). Within commercial DBMSs predictive functionality is usually implemented as customized functions using proprietary languages [1, 25] and thus, cannot be utilized with other relational operators. In contrast, Paris et. al. [26] developed a formal definition of a forecast operator and explored the integration of forecast operators with standard relational operators.

Transparent and Automatic Query Processing The processing of forecast queries should be done transparently and automatically by the database system. This includes automatic creation or reuse of forecast models for given forecast queries as well as automatic maintenance of materialized forecast models. For example, within the Fa system [10] an incremental approach was proposed to automatically build ad-hoc models for multi-dimensional time series, where more attributes are added to the model in successive iterations.

Efficient Query Processing Complex forecast models or large amount of time series data might lead to long execution time of forecast queries. Thus, optimization techniques are required that efficiently process such forecast queries. Ge and Zdonik [16] proposed an I/O-conscious skip list data structure for very large time series. In addition, techniques to efficiently reuse models exploiting multi-dimensional data have already been developed [2, 13, 14].

Efficient Update Processing Finally, a continuous stream of new time series values requires efficient maintenance of computed models. General [5] and model-specific [28] techniques to determine when a model requires maintenance have been proposed. In addition, existing approaches speed up the computation process itself by parallelization [6] or by using previous model parameters [7].

To summarize, existing research papers already identified individual aspects of the requirements in the area of time series forecasting. However, no general architecture including all these requirements and exploiting existing optimization approaches has been described so far. In the next section, we present a general architecture that addresses this issue.

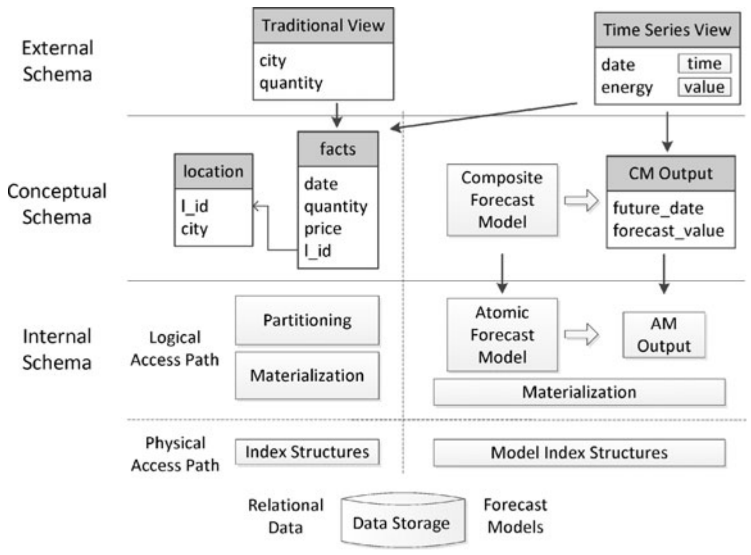
3 Forecasting in Database Management Systems

The basic idea underlying our forecasting DBMS approach is to develop a DBMS architecture that specifically supports and is optimized for the execution of *forecast queries*, i.e., queries that involve forecasted values. For this purpose, we decided to base our approach on the standard ANSI/SPARC architecture [24] and enhance it with specific forecasting components. In particular, we propose specific changes and additions on all three levels of the ANSI/SPARC architecture to allow a transparent and efficient end-to-end execution of forecast queries. Figure 2 illustrates the ANSI/SPARC architecture as well as our proposed additions. Our changes on the external schema level mainly comprise the definition of a special *Time Series View* that ensures a representation of data in combination with time. On the conceptual schema level, we define a *Composite Forecast Model* that is a conceptual representation of a concrete (atomic) forecast model. Composite models can define a forecast model composition, meaning that the forecast model is decomposed into multiple individual models. With this approach, we achieve a higher accuracy by intelligently model compositions and reduce maintenance costs by reusing models in multiple compositions. The internal schema is divided into the Logical Access Path [30] and the Physical Access Path. The Logical Access Path contains the *Atomic Forecast Model* that is a concrete realization of a forecast model defining the model type and characteristics. Further on this level, we propose a possible materialization of forecast models and forecasting results to quickly provide results for frequently repeating forecast queries. On the Physical Access Path, Fischer et al. [14] suggested to use special *Forecast Model Index Structures* that further increase the efficiency of forecast queries. Additionally, specific time series data index

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum" 13 (1), S. 45–53. ISSN: 1610-1995
http://dx.doi.org/10.1007/s13222-012-0108-4

Fig. 2 Extension of 3-layer schema architecture



structures [16] can be exploited to speed up time series access. In the following, we describe our enhancements for the ANSI/SPARC architecture in more detail and show the application of our concept in a real world example.

3.1 System Architecture Details

External Schema The external schema in the ANSI/SPARC architecture consists of user-defined data views, which can be seen as virtual tables storing the results of specific queries. A view can comprise attributes of multiple tables as well as pre-defined aggregations or calculation results. Forecasts are typically calculated on time series data, meaning a sequence of discrete values measured successively over time. To allow forecast queries in a database system, we define a special type of a regular view that ensures the representation of data as time series. Thus, the *Time Series View* is composed of an obligatory time attribute containing discrete points in time in its natural order and one or multiple attributes exhibiting measurements at these specified moments. Optionally, these attributes can be tagged with forecasting-specific meta data, which, for example, indicates whether an attribute represents a dependent variable to be forecasted or an independent variables such as *external influence* having an impact on the main variable. Typically, there is one main variable and many external influences. An example query defining a *Time Series View* is denoted as:

```
CREATE TIMESERIESVIEW tv1 AS
SELECT date as TIME, energy as VALUE,
temperature as INFLUENCE
FROM measurements
ORDER BY TIME;
```

A time series view can represent both historical and forecasted values of a time series. Typically, after its creation only historic values are contained, but as soon as a query requests future values, these values are forecasted and added to the view accordingly. For predicted values, the view also contains further information such as the standard deviation or confidence intervals, which clearly distinguish future values from historic values. Once real values are available, they replace the forecasted values. The time series view is typically defined by a user or an application, and, once defined, it is automatically managed by the DBMS (e.g., by automatically calculating forecasts or adding new values). The view can be queried in an ad-hoc fashion at any time and can be referenced by any other regular view or time series view. In some cases, as explained later, a time series view is generated automatically by the DBMS as the result of a forecast model decomposition.

Conceptual Schema The conceptual organization of the data in an ANSI/SPARC compliant DBMS is defined in the conceptual schema level. This level includes a data schema that describes available entities, their relationship and contained attributes and can be seen as an abstraction from the logical and physical data representation. Likewise, *Composite Forecast Models* are defined as a conceptual abstraction from concrete (atomic) forecast models and thus, it can be seen as some kind of transparency layer. In a simple case, a conceptual model directly refers to a single atomic forecast model from the internal schema, representing a simple direct forecast, e.g., energy production of a single solar panel. In a more complex case, composite forecast models can also describe a hierarchical forecast composition. When forecasting the energy consumption of Germany, for example, the forecasting can be decomposed into forecasts of the energy consumption for all German states, or further down in the hierarchy, the energy consumption of all German cities. For this purpose, the composite forecast model can define a hierarchical forecast composition referring further composite

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum" 13 (1), S. 45–53. ISSN: 1610-1995
http://dx.doi.org/10.1007/s13222-012-0108-4

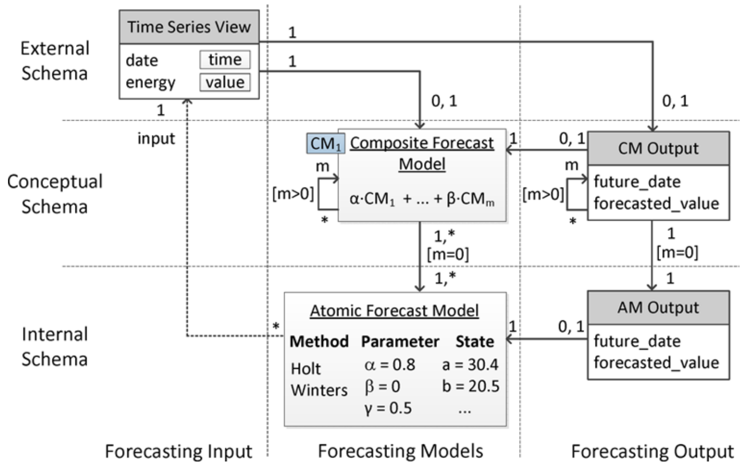


Fig. 3 Relationship between forecasting components

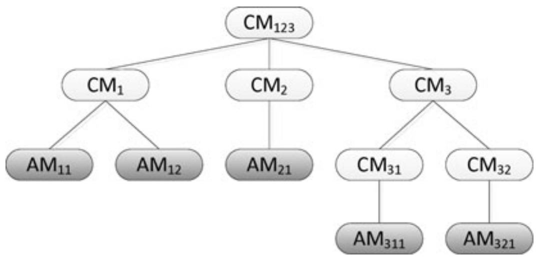


Fig. 4 Example model hierarchy

forecast models on multiple hierarchy levels. The final forecast is later calculated by aggregating the single results of the referenced atomic forecast models according to the defined hierarchical composition. As illustrated in Fig. 3, each composite forecast model in the hierarchy can either reference multiple child composite forecast models or, on the leaf level, ultimately refer to atomic forecast models defined in the internal schema layer of the ANSI/SPARC architecture. Figure 4 shows an example of such a forecast model composition hierarchy. It is important to note that the automatic determination of forecasting compositions is a complex task with many prerequisites and constraints. For now we assume that compositions are pre-defined by the database administrator that is aware of the database schema and the available data. We discuss the automatic composition creation separately in Sect. 3.2.

Besides the definition of forecasting compositions, typically, one composite forecast model references only one atomic forecast model (see Fig. 3). However, for the sake of forecasting accuracy it is also possible to employ ensemble forecasting, where multiple atomic models of different types and with different parameter combinations are executed in parallel (e.g., [29, 31]). Afterwards, the results are combined in a weighted linear combination, where the most accurate model gains the highest weight. In this case, a single composite model might refer to multiple atomic models.

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 ("CM Output"), which is a special table complying to the same rules as the time series view.

Internal Schema The logical and the physical data access paths are defined in the internal schema of an ANSI/SPARC architecture. Logical access paths refer to aspects like partitioning and materialization, while 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 (1) an *input*, (2) the *forecast model type*, (3) *forecast model parameters* and (4) the current *forecast state* (see Fig. 3). Here, the input is the data as defined in the associated time series view, referenced through the connected composite forecast model. The forecast model type (e.g., exponential smoothing) is chosen from a *forecast model catalog* that represents all forecast model types available in the DBMS and is pre-defined with respect to the application domain and the common data characteristics. The chosen forecast model type determines the forecast model characteristics (e.g., number of lags) and defines the parameters of the forecast model. When creating an instance of an atomic forecast model the parameters are estimated using local or global general purpose optimization algorithms. This estimation involves a large number of simulations and thus, typically is very time consuming. These algorithms are required to be part of a forecast-enabled DBMS. Finally, the output of the atomic forecast model—predicted future values—are stored in a special data structure called the *atomic forecast model output* ("AM Output"). This table is composed of at least a time column and exactly one value column that contains the forecasted values. Optionally, additional attributes might be included in the atomic forecast model output (e.g., prediction intervals).

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum" 13 (1), S. 45–53. ISSN: 1610-1995
http://dx.doi.org/10.1007/s13222-012-0108-4

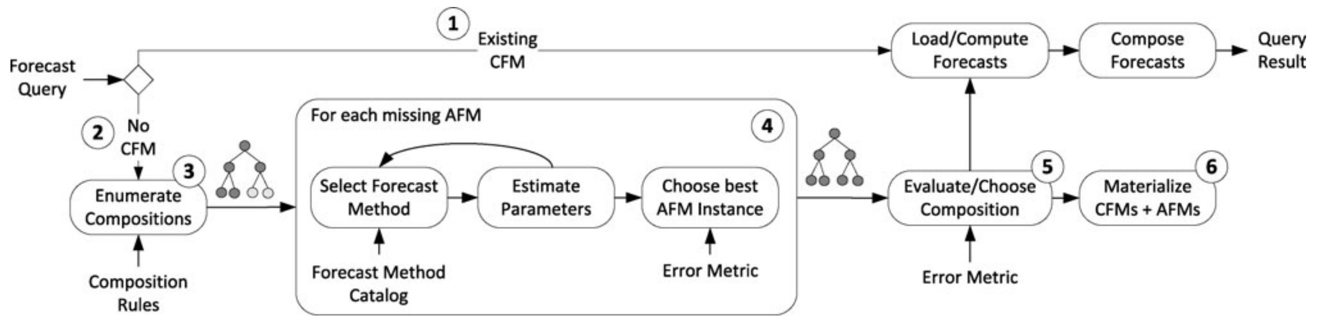


Fig. 5 Processing Forecast Queries

Similarly to materialized views, composite and atomic forecast models can be computed on the fly or materialized for faster query response times. Materialized forecast models store the forecast model parameters and the forecast model state. This especially avoids the very time consuming estimation of the forecast model parameters and thus, allows a very fast provisioning of forecasting results. As a result, the execution time of forecast queries is greatly reduced. A further reduction is possible when directly materializing the forecasting results, i.e., the forecast model output. Similar to materialized views, materialized forecast models require maintenance after each appended time series value. This includes either the simple update of the forecast (when the model is still accurate) or a more expensive parameter re-estimation (when the model violates the accuracy constraints). The maintenance of the materialized forecast models can be conducted asynchronously to the execution of forecast queries, which allows for fast forecast query execution at all times. For the physical access paths of the internal schema, model index structures can be used that allow an efficient storage and search for materialized composite forecast models and connected atomic forecast models [14]. Such index structures ensure efficient updates (or invalidations) of atomic forecast models on time series updates as well as efficient access to instantiated atomic forecast models and their outputs during the execution of forecast queries. Additionally, classical data index structures are enhanced to allow an efficient processing of time series data. First, time index structures are necessary to ensure access of the time series values in a timely subsequent order as it is required for forecast models. Second, more advanced indexing structures like skip-lists [16] can be utilized or similarity indexes, which would further increase the processing efficiency and decrease the forecast query response times.

To sum up, we have discussed necessary conceptual extensions to the traditional ANSI/SPARC architecture in order to enable forecasting within a DBMS. For further readings and details on the implementation of such an architecture we refer to [15].

3.2 Processing Forecast Queries

Based on the introduced conceptual architecture, we now discuss the processing of a forecast query. We first give a high level overview of the basic steps in forecast query processing (Sect. 3.2.1) and then traverse the process in more detail using an example from the use case of energy forecasting (Sect. 3.2.2).

3.2.1 Overview

We distinguish two main cases to process a given forecast query (Fig. 5). First, if a suitable composite forecast model exists, the forecasts are computed or loaded (either model parameters or materialized forecast values) for each atomic forecast model within the given composite model and compose the final forecasts according to the stored composition rule ((1) in Fig. 5). Second, if no composite forecast model is available, two choices arise. We can either return an error to the user or we can compute a composite forecast model on the fly ((2) in Fig. 5). In the latter case, we first enumerate different composition alternatives using a *composition rule catalog* or composition advisor (3). Such a catalog might store meta data describing the hierarchical dependencies in the data that can be used as composition rules. If no such rule catalog is available, we do not create a forecasting composition, but create a single composite forecast model that attaches exactly one atomic forecast model to the given time series view. For all found composition alternatives, we then create all missing atomic forecast models that do not already exist in the database (4). Such an atomic forecast model is created by empirically evaluating different forecast methods that are stored in the *forecast method catalog* and choosing the best one or by employing forecast model ensembles. Finally, after all atomic forecast models have been created, we choose the composition approach with the lowest error and calculate the query result (5). The previously described forecasting process, including the model composition creation, is conducted transparently to the user. The user simply requests a forecast for the user-defined time series view and

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum" 13 (1), S. 45–53. ISSN: 1610-1995
http://dx.doi.org/10.1007/s13222-012-0108-4

Table 1 Energy consumption relation ("measurements")

Date	Group	Amount
2012-01-01 09:00	1 (households)	200
2012-01-01 09:00	2 (small industries)	500
2012-01-01 09:00	3 (large industries)	1500
2012-01-01 09:15	1	250
2012-01-01 09:15	2	525
2012-01-01 09:15	3	1600
...	...	...

the system automatically decides upon the necessary steps to provide the results. Finally, to avoid expensive parameter estimation for future queries, the system might choose to materialize the final composite and corresponding atomic forecast models, including the model parameters and model state (6). In the following, we further detail this process of automatic composite forecast model creation.

3.2.2 *Energy Forecasting Example*

In the energy domain, the availability of accurate forecasts of future electricity consumption and production is a prerequisite for the balancing of energy demand and supply and thus ensuring the stability and efficiency of the energy grids. Forecasts are produced on different hierarchy levels that exist in the physical energy grid, e.g., street, district, city, or country level. In the following example, we assume a utility company collecting metering data from households, small, and large industrial consumers and using this metering data to forecast a total demand to be bought on the wholesale electricity market for the next day. We now show how our enhanced DBMS architecture can be utilized to forecast the total demand for this scenario.

Suppose, energy consumption data of different consumer groups (households, small/large industries) is stored like shown in Table 1. Initially, a *time series view* over this data, aggregating energy consumption measurements, is created:

```
CREATE TIMESERIESVIEW tsConsTotal AS
 SELECT Date as TIME, SUM(Amount) as VALUE
 FROM measurements
 WHERE Group BETWEEN 1 AND 3
 GROUP BY TIME ORDER BY TIME
```

Now, suppose a user submits the following query over this time series view:

```
SELECT time, value
 FROM tsConsTotal
 WHERE time in (yesterday(), tomorrow())
```

The system seamlessly determines if the query involves forecasting (accesses future values) or not. As the user query in our example requests the total energy consumption for the next day, the system automatically triggers a search for a corresponding *composite forecast model* in the DBMS. In our example, no corresponding model is found and the system seamlessly creates a new composite forecast model. The system has many different alternatives to create this model and it might spend some time on choosing an alternative, which offers the best forecasting accuracy. Suppose the composition rule catalog outputs a very simple composition rule that suggests to aggregate the forecasts of individual consumer groups (1, 2, and 3) to retrieve the overall energy consumption forecast. Further assume that two composite forecast models, CM_2 and CM_3, already exist in the database (as they are used by other time series views previously defined by a user) to forecast the consumption of small and large industries. To evaluate this composition rule, the system needs to create an additional composite forecast model for energy consumption of private households CM_1. This model CM_1 requires an input, defined by the following automatically generated time series query:

```
SELECT Date as TIME, Amount as VALUE
 WHERE Group = 1 (i.e., households)
 FROM measurements
 GROUP BY TIME ORDER BY TIME
```

When creating CM_1, the system creates underlying atomic forecast model AM_1 and empirically evaluates the forecast methods listed in the *forecast method catalogue*. For AM_1, in our example, the forecast method *Triple Seasonal Exponential Smoothing* [32] is chosen as the most accurate solution for forecasting household consumption data for the specific data set. Then, the models CM_1, CM_2, and CM_3 are composed using the following rule $\alpha \cdot \mathrm{CM}_1 + \beta \cdot \mathrm{CM}_2 + \gamma \cdot \mathrm{CM}_3$. α, β and γ are the weights of the linear combination describing the impact of the respective consumer groups. Typically, the weights reflect the share of each consumer group on the total consumption, which is computed from the history of the corresponding time series.

The accuracy of this composition rule might be compared to other composition rules (e.g., create only one composite model for the overall energy consumption time series `tsConsTotal`), which also require the creation of missing atomic forecast models. Finally the composition rule producing the most accurate forecasts is chosen (the aggregation of individual groups in our example) and the output to the query is obtained by aggregating the forecast values from the outputs of CM_1, CM_2, and CM_3 (see "CM Output" in Fig. 2) at respective time stamps. Then, the output is merged with historical values of `tsConsTotal` from yesterday and the result set is returned to the user.

Finally, the system might choose to materialize the composite and corresponding atomic forecast models including, for our example, CM_1 and the parameters of AM_1, to speed up the processing of future queries. Additionally, the forecasting result might be materialized.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

4 Research Topics and Challenges

Realizing a forecast-enabled DBMS, which takes advantage of our proposed architecture, is challenging. Some aspects where already addressed by individual research papers (see Sect. 2). In this section, we focus on remaining challenges that either nobody has considered yet or that arise due to our novel architecture.

External Schema Queries of different types have to be supported, processed, and optimized to take advantage of the models stored within a DBMS. These include traditional ad-hoc as well as reoccurring and continuous forecast queries. Existing SQL extensions [15] might be further refined to allow seamless querying of time series data that does not require the specification of a `FORECAST` keyword, e.g., by restricting the time in the where clause of a query (`WHERE time IN (now(), tomorrow())`) Now the system additionally needs to detect if a certain time series query involves forecasting or just demands the history of the time series. In addition, query constraints might be specified on the desired accuracy or runtime. This requires anytime or online approaches that progressively provide better forecast results over time.

Conceptual Schema Each forecast query on a time series view requires either the use of an existing composite forecast model or a new model needs to be built. Large databases might contain millions of time series [2], requiring efficient strategies to search and build composite forecast models. The decomposition of composite forecast models into a hierarchy of multiple composite forecast models can be either done manually or automatically. Automatic approaches face two main challenges. First, they need to determine what decompositions are possible and, second, they need to choose the best decomposition. Suitable decompositions might be given by the database administrator in terms of composition rules, derived automatically from the data (e.g., by using foreign-key relationships) or given by meta data like data cubes or data hierarchy information. The determination of the most accurate decomposition is quite a hard problem as the number of possible decomposition might be very high and as the accuracy of a decomposition cannot be determined without actually building all concerning models [11]. First approaches in this area [13, 19] are only suitable for a small number of time series. Additionally, the system might transparently maintain composite forecast models. Thus, the system can automatically switch to a new composition if it leads to a higher forecast accuracy.

Internal Schema—Logical Access Path As forecast queries should be processed transparently, atomic models have to be chosen and created automatically. Due to the large variety of possible models and parameterizations, this process of model identification is challenging. Domain-specific model types can reduce the search space by including only models from the given domain. Automatization approaches are required that automatically select the "best" model for a time series. First approaches automatize the model selection process for ARIMA [20] or Exponential Smoothing [21] models. Through the usage of ensemble models, i.e., combination of several individual atomic models, the robustness and accuracy of such approaches might be improved. Once one or several models have been selected, model parameters need to be estimated, which is very time consuming due to a parameter search space that increases exponentially with the number of parameters. Sophisticated approaches are necessary that decrease the estimation runtime (e.g., by parallelization) as well as increase the probability of finding a global optima.

To avoid expensive model creation, atomic forecast models might be materialized so they can be used over and over again. However, due to evolving time series in many domains, materialized models require maintenance in form of parameter reestimation. Research in this area focuses on two main challenges, first, when to reestimate parameters and, second, how to speed up the reestimation itself. Existing research papers already address both challenges (as discussed in Sect. 2) and might be extended and improved.

Due to expensive model maintenance, materialized atomic forecast models have to be selected carefully. The question arises how to intelligently reuse models in order to keep maintenance costs as low as possible while enabling high accuracy (e.g., one atomic forecast model might be used in several composite forecast models). Such a configuration of atomic forecast models might be chosen offline by the user or online using automatic approaches, which, in addition to model maintenance, requires continuous evaluation and adaption.

Internal Schema—Physical Access Path. Finally, specific index structures on time series data or forecast models might be used to speed up model creation, usage and maintenance—as discussed in Sect. 3.1. These initial approaches might be improved by more advanced index structures, either for a specific model type or for the general case. In addition, partitioning the data with index structures might additionally enable parallelization approaches that parallelize within or between parameter estimators and models. One such parallelization approach was proposed for a energy-domain-specific forecast model [6] and might be extended to other model types.

Multiple research aspects remain on all three layers of the ANSI/SPARC architecture and open up many interesting research directions. In contrast to traditional query processing, all forecasting relevant topics (e.g., selection, estimation, maintenance) need to cope with a two dimensional

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Datenbank-Spektrum" 13 (1), S. 45–53. ISSN: 1610-1995
http://dx.doi.org/10.1007/s13222-012-0108-4

optimization objective–forecast accuracy versus runtime of forecast query processing.

5 Conclusions

We addressed a current research trend that deals with the integration of statistical methods into data management systems. Within this article, we explicitly discussed the integration of time series forecasting, which allows for declarative, transparent and efficient forecast queries. We introduced a generic forecasting architecture that is based on the traditional ANSI/SPARC architecture and divides the forecasting components into different abstraction layers. Although different applications need to support different query and model types, they build upon the same general forecasting architecture. Thus, similar optimization opportunities and challenges arise that need to be exploited in future work.

References

1. PredictTimeSeries–Microsoft SQL server 2008 books online (2012). http://msdn.microsoft.com/en-us/library/ms132167.aspx
2. Agarwal D, Chen D, ji Lin L, Shanmugasundaram J, Vee E (2010) Forecasting high-dimensional data. In: SIGMOD conference, pp 1003–1012
3. Böhm M, Dannecker L, Doms A, Dovgan E, Filipic B, Fischer U, Lehner W, Pedersen TB, Pitarch Y, Siksnys L, Tusar T (2012) Data management in the MIRABEL smart grid system. In: EDBT/ICDT workshops, pp 95–102
4. Cohen J, Dolan B, Dunlap M, Hellerstein JM, Welton C (2009) MAD skills: new analysis practices for big data. Proc VLDB Endow 2(2):1481–1492
5. Dannecker L, Böhm M, Lehner W, Hackenbroich G (2011) Forcasting evolving time series of energy demand and supply. In: ADBIS, pp 302–315
6. Dannecker L, Böhm M, Lehner W, Hackenbroich G (2012) Partitioning and multi-core parallelization of multi-equation forecast models. In: SSDBM, pp 106–123
7. Dannecker L, Schulze R, Böhm M, Lehner W, Hackenbroich G (2011) Context-aware parameter estimation for forecast models in the energy domain. In: SSDBM, pp 491–508
8. Das S, Sismanis Y, Beyer KS, Gemulla R, Haas PJ, McPherson J (2010) Ricardo: Integrating R and hadoop. In: SIGMOD conference, pp 987–998
9. Deshpande A, Madden S (2006) MauveDB: supporting model-based user views in database systems. In: SIGMOD conference, pp 73–84
10. Duan S, Babu S (2007) Processing forecasting queries. In: VLDB'07, pp 711–722
11. Dunn D, Williams W, DeChaine T (1976) Aggregate versus subaggregate models in local area forecasting. J Am Stat Assoc 71:68–71
12. Faerber F, Cha SK, Primsch J, Bornhoevd C, Sigg S, Lehner W (2011) SAP HANA database—data management for modern business applications. SIGMOD Rec 40:45–51
13. Fischer U, Böhm M, Lehner W (2011) Offline design tuning for hierarchies of forecast models. In: BTW, pp 167–186
14. Fischer U, Rosenthal F, Böhm M, Lehner W (2010) Indexing forecast models for matching and maintenance. In: IDEAS, pp 26–31
15. Fischer U, Rosenthal F, Lehner W (2012) F2DB: the flash-forward database system. In: ICDE, pp 1245–1248
16. Ge T, Zdonik SB (2008) A skip-list approach for efficiently processing forecasting queries. Proc VLDB Endow 1(1):984–995
17. Gooijera JGD, Hyndman RJ (2006) 25 years of time series forecasting. Int J Forecast 22:443–473
18. Große P, Lehner W, Weichert T, Färber F, Li WS (2011) Bridging two worlds with RICE integrating R into the SAP in-memory computing engine. Proc VLDB Endow 4(12):1307–1317
19. Hyndman RJ, Ahmed RA, Athanasopoulos G, Shang HL (2011) Optimal combination forecasts for hierarchical time series. Comput Stat Data Anal 55(9):2579–2589
20. Hyndman RJ, Khandakar Y (2008) Automatic time series forecasting: the forecast package for R. J Stat Softw 27:1–22
21. Hyndman RJ, Koehler AB, Snyder RD, Grose S (2000) A state space framework for automatic forecasting using exponential smoothing methods. Int J Forecast 18:439–454
22. Jeung H, Yiu ML, Zhou X, Jensen CS (2010) Path prediction and predictive range querying in road network databases. VLDB J 19(4):585–602
23. Koc ML, Ré C (2011) Incrementally maintaining classification using an RDBMS. Proc VLDB Endow 4(5):302–313
24. Lehner W (2003) Datenbanktechnologie für Data-Warehouse-Systeme. Konzepte und Methoden. dpunkt
25. Oracle (2012) Oracle OLAP DML reference: FORECAST–DML statement
26. Parisi F, Sliva A, Subrahmanian VS (2011) Embedding forecast operators in databases. In: Proceedings of the 5th international conference on scalable uncertainty management (SUM'11), pp 373–386
27. Ramanathan R, Engle R, Granger CWJ, Vahid-Araghi F, Brace C (1997) Short-run forecasts of electricity loads and peaks. Int J Forecast 13(2):161–174
28. Rosenthal F, Lehner W (2011) Efficient in-database maintenance of ARIMA models. In: SSDBM, pp 537–545
29. Rosenthal F, Volk PB, Hahmann M, Habich D, Lehner W (2009) Drift-Aware ensemble regression. In: Proceedings of the 6th international conference on machine learning and data mining in pattern recognition (MLDM'09), pp 221–235
30. Roussopoulos N (1982) The logical access path schema of a database. IEEE Trans Softw Eng 8:563–573
31. Sánchez I (2008) Adaptive combination of forecasts with application to wind energy. Int J Forecast 24(4):679–693
32. Taylor JW (2009) Triple seasonal methods for Short-term electricity demand forecasting. Eur J Oper Res 204:139–152
33. Winter R, Kostamaa P (2010) Large scale data warehousing: trends and observations. In: ICDE, p 1
34. Xu B, Wolfson O (2003) Time-Series prediction with applications to traffic and moving objects databases. In: Proceedings of the 3rd ACM international workshop on data engineering for wireless and mobile access (MobiDe'03), pp 56–60

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden