Ulrike Fischer, Frank Rosenthal, Wolfgang Lehner

Sample-based forecasting exploiting hierarchical time series

Erstveröffentlichung in / First published in:

IDEAS '12: 16th International Database Engineering & Applications Symposium, Prague 08.08. – 10.08.2012. ACM Digital Library, S. 120–129. ISBN 978-1-4503-1234-9

DOI: https://doi.org/10.1145/2351476.2351490

Final edited form was published in "IDEAS '12: 16th International Database Engineering & Applications Symposium. Prague 2012", S. 120–129, ISBN 978-1-4503-1234-9
https://doi.org/10.1145/2351476.2351490

Sample-Based Forecasting Exploiting Hierarchical Time Series

Ulrike Fischer, Frank Rosenthal, Wolfgang Lehner
Dresden University of Technology
Database Technology Group
01062 Dresden, Germany
{firstname.lastname}@tu-dresden.de

ABSTRACT

Time series forecasting is challenging as sophisticated forecast models are computationally expensive to build. Recent research has addressed the integration of forecasting inside a DBMS. One main benefit is that models can be created once and then repeatedly used to answer forecast queries. Often forecast queries are submitted on higher aggregation levels, e.g., forecasts of sales over all locations. To answer such a forecast query, we have two possibilities. First, we can aggregate all base time series (sales in Austria, sales in Belgium ...) and create only one model for the aggregate time series. Second, we can create models for all base time series and aggregate the base forecast values. The second possibility might lead to a higher accuracy but it is usually too expensive due to a high number of base time series. However, we actually do not need all base models to achieve a high accuracy, a sample of base models is enough. With this approach, we still achieve a better accuracy than an aggregate model, very similar to using all models, but we need less models to create and maintain in the database. We further improve this approach if new actual values of the base time series arrive at different points in time. With each new actual value we can refine the aggregate forecast and eventually converge towards the real actual value. Our experimental evaluation using several real-world data sets, shows a high accuracy of our approaches and a fast convergence towards the optimal value with increasing sample sizes and increasing number of actual values respectively.

Categories and Subject Descriptors

H.2.4 [**Systems**]: Query processing; G.3 [**Probability and Statistics**]: Time series analysis

1. INTRODUCTION

Advanced data analysis in data-warehouse systems involves increasingly sophisticated statistical methods that go well beyond the rollup and drilldown of traditional BI [9]. In this context, time series forecasting is an important instrument as it is crucial for decision making in many domains. As a running example, we refer to a market research company that collects monthly sales of different products (e.g., audio devices) delivered by different retailers. With this data they generate aggregated reports over certain products and provide forecasts for the next month to their customers, e.g., manufacturers. These reports form the basis of economic decisions, such as planning of production batches.

Reasonable forecasts require the specification of a stochastic model that captures the dependency of future on past values. We will refer to such models as *forecast models*. The creation of forecast models is typically computationally expensive, often involving numerical optimization schemes to estimate model parameters. Once a model is created and parameters are estimated, it can efficiently be used over and over again to forecast future values of the time series. As new data arrives, the forecast model might require *maintenance* in form of parameter re-estimation, which is computationally expensive as well as most parameters can not be maintained incrementally.

Current research focuses on the integration of time series forecasting and prediction in general inside a DBMS [5, 10, 13]. These approaches allow for improved performance and additional functionality inside the database. Consider the following simple forecast query that retrieves forecasts of audio devices for the next month:

```
SELECT   orderdate, SUM(sales)
FROM     facts
WHERE    product='audio devices'
GROUP BY orderdate
FORECAST current_date + interval '1 month'
```

One could answer such a query by computing a forecast model on the fly, which, however, leads to long processing times due to expensive model creation. Thus, forecast models are precomputed and stored within the DBMS. In addition, others queries can benefit from forecasts based on these models. A main challenge is the determination of a good set of models to store in the database that reduce model maintenance costs and achieve high forecast accuracy. Especially aggregation hierarchies open up the possibility of reducing the number of forecast models by applying derivation schemes [4, 12].

In aggregation hierarchies, each unique combination of dimension attributes (e.g., product name, location) forms a *base time series*. Queries often request forecasts on aggregation of these time series. For example, monthly sales of

Final edited form was published in "IDEAS '12: 16th International Database Engineering & Applications Symposium. Prague 2012", S. 120–129, ISBN 978-1-4503-1234-9
https://doi.org/10.1145/2351476.2351490

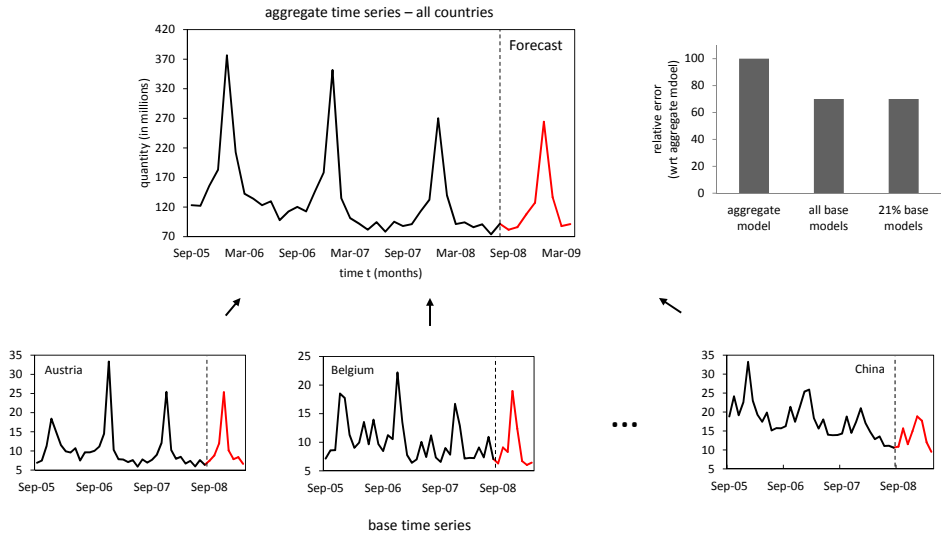


Figure 1: Aggregation of Monthly Sales of Audio Devices.

audio devices might be recorded according to different locations (bottom of Figure 1). Our example aggregation forecast query introduced before requests the aggregate forecast of audio devices over all locations (top of Figure 1). We have two possibilities to provide the result:

- *Aggregate → Forecast:* First, we can aggregate all base time series and build *one* model for the aggregate time series. The aggregate forecast is then calculated using this model.
- *Forecast → Aggregate:* Second, we can build models for *all* base time series and produce forecasts on base level. Those forecasts are then aggregated to retrieve the aggregate forecast.

The second possibility might lead to a higher accuracy of the aggregate forecast as it might be easier to find good models for more fine granular elements. This was studied and examined on different data sets in statistical and specific forecasting literature [14].

However, as model creation and maintenance is computationally expensive, it is not always possible to create and store all base models. Thus, an aggregate model with a lower accuracy is used instead. However, we actually do not need all base models, a sample of base models can still outperform the associated aggregate model. In the top right corner in Figure 1, we display an initial experiment using the sales data set of audio devices. Our baseline is the forecast error of an *aggregate model* created with the aggregate time series over all locations. We can observe that the error is decreased by 25% if we aggregate the forecasts of all *base models* instead. However, if we just use a 21% sample of base models, we achieve pretty much the same accuracy as using all models. Thus, we need to create and maintain less than a quarter of the models, but we do not loose any accuracy compared to using the aggregate model.

Following this approach, we can further improve the aggregate forecast if some of the real data is already available. For example, it is quite common that the market research company retrieves sales of different retailers at different point in times. An successful retailer might already have reported his sales in his location, while another retailer has not provided any data yet. Thus, the new real values of the base time series arrive asynchronously. However, customers wish to retrieve aggregated statistics at all times. Although, we can not yet provide the real aggregate, we can *refine* the aggregate forecast with the available real values and eventually converge towards the real aggregate value.

Contributions and Outline We first give an overview of our system architecture and the involved components (Section 2). Then, in Section 3, we propose different estimators to calculate an overall aggregate forecast from a sample of base forecasts. This aggregate forecast can be further refined by including already available real values (Section 4). Last, in Section 5, we shortly discuss how to configure a good sample of base time series. Our experimental study (Section 6) shows a high forecast accuracy with just a small sample of base time series and fast convergence towards the optimal value with increasing sample size and increasing number of real values respectively. Finally, we survey related work in Section 7 and conclude in Section 8.

2. SYSTEM ARCHITECTURE

Figure 2 shows our high-level system architecture. Time series are stored within the DBMS, where we refer to non-aggregated time series as *base time series*. A base time series X_i contains a chronologically ordered series of observations over time $x_{i,1}, x_{i,2}, ..., x_{i,t}$, where i clearly identifies the base time series. The *model pool* stores pre-built forecast models M_i created over base time series X_i. The type of the forecast model (e.g., exponential smoothing) needs to be chosen by a domain expert or using an heuristic algorithm [22]. The decision for which base time series X_i to build and store a model in the pool can be done manually [4] or by using an empirically algorithm that tries different configurations and chooses the best one [12]. In Section 5, we shortly discuss how to design a good pool of forecast models.

An aggregation forecast query requires the one-step ahead aggregated forecast $\hat{y}_{t+1}$ over an arbitrary number of base time series at time t. For sake of simplicity of the presentation, we restrict the aggregate forecast to the *sum* and we

Final edited form was published in "IDEAS '12: 16th International Database Engineering & Applications Symposium. Prague 2012", S. 120–129, ISBN 978-1-4503-1234-9
https://doi.org/10.1145/2351476.2351490

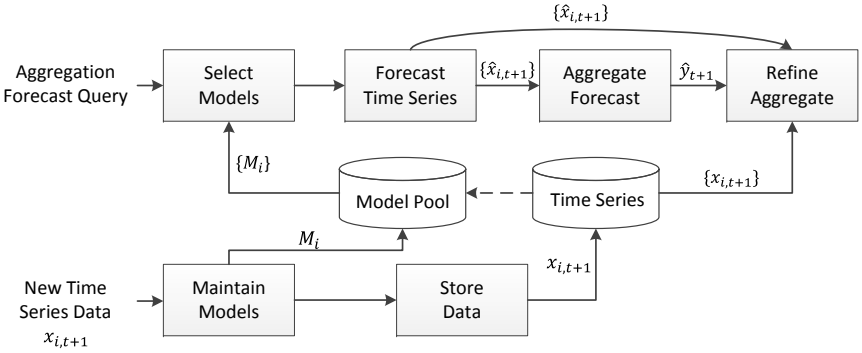


Figure 2: System Architecture.

only consider *one-step ahead* forecasts.

In general, it is processed as follows (top of Figure 2): We first *select* all base models $\{M_i\}$ that are available and can be used to answer that query. With each model we calculate the *forecast* value for the corresponding base time series X_i, which we denote as $\hat{x}_{i,t+1}$. This calculation of forecasts can be done very quickly as there is no need to access the base data. We then estimate the *aggregate forecast* $\hat{y}_{t+1}$ over the given base forecasts (Section 3). Finally, we check if some of the real time series values $x_{i,t+1}$ are already available at the current point in time t. If so, we further *refine* the aggregate forecast using the error between the base forecast values as well as the real time series values (Section 4).

As new actual time series values arrive $x_{i,t+1}, x_{i,t+2}, ...$ we need to *maintain* our model (bottom of Figure 2): First, this requires updating the model to the current state of the time series. Second, we might need to adapt the parameters of the model. As most parameters can not be calculated incrementally, parameter re-estimation is as expensive as model creation itself. Therefore, it is not just important to reduce the number of stored models in the pool but also to reduce the number of parameter re-estimations of a single model. Different strategies have been proposed to trigger parameter re-estimation, like threshold-based, update-based [16] or on-demand [25]. Here, we will use the simple time-based strategy that triggers parameter re-estimation every x inserts. This is one of the most robust strategies. Finally, we *store* the new time series values in the DBMS.

We now further detail the main components of our system, which is the aggregation of the forecast values (Section 3), the refinement of the forecasts with new data (Section 4) and the initial design of the forecast model pool (Section 5).

3. AGGREGATION

In this section, we discuss how to obtain an overall aggregate forecast from a given sample of base forecast values. We first introduce the mathematical fundamentals of estimating the sum of a population (Subsection 3.1) and discuss uniform estimation (Subsection 3.2). Then, we improve this approach by using the history of the time series values (Subsection 3.3).

3.1 Aggregation Basics

Consider N base time series $\{X_i\}$ and corresponding forecast values $\{\hat{x}_{i,t+1}\}$, where $1 \leq i \leq N$. An aggregation forecast query wants to retrieve the aggregate forecast $\hat{y}_{t+1}$, which can be calculated by

$$\hat{y}_{t+1} = \sum_{i=1}^{N} \hat{x}_{i,t+1}. \tag{1}$$

Our goal is to estimate the forecast value $\hat{y}_{t+1}$ of the aggregate time series from a sample of base time series $S = \{X_i\}$ of size $n \leq N$. Horvitz and Thompson [21] introduced an estimator that can be used with any probability design without replacement. The estimate of the sum is given by

$$\hat{y}_{t+1} = \sum_{X_i \in S} \frac{\hat{x}_{i,t+1}}{\pi_{i,t+1}}, \tag{2}$$

where $\pi_{i,t+1}$ denotes the first-order inclusion probability of $\hat{x}_{i,t+1}$. The higher the probability of inclusion, $\pi_{i,t+1}$, of a unit i to the sample, the less weight the corresponding response $\hat{x}_{i,t+1}$ is given. Therefore, each value is scaled-up or "expanded" by its inclusion probability. The main challenge is to choose good inclusion probabilities.

3.2 Uniform Aggregation

If there is no additional information available, we choose uniform probability of inclusion. Therefore, we set $\pi_{i,t+1} = n/N$ for all i and the aggregate forecast is estimated by

$$\hat{y}_{t+1} = \frac{N}{n} \sum_{X_i \in S} \hat{x}_{i,t+1}. \tag{3}$$

However, uniform estimation might lead to a strong under- or overestimation of the aggregate value, because different base forecasts might contribute differently to the aggregate forecast. For example, if we have a 50% sample, we would scale the sum by two. However, our sample might contain many high forecast values, which are in total much higher than half the aggregate forecast. Thus, we would strongly overestimate the aggregate forecast. As we are in a time series context, we can additionally use the information in the history of the time series to provide better estimates. The key idea is to use the historical ratio of the base time series to the overall aggregate series as inclusion probabilities.

3.3 Aggregation with Historical Ratios

We denote the *historical ratio* $d_{i,t+1}$ of a base value $x_{i,t+1}$ and an aggregate time series value y_{t+1} as $d_{i,t+1} = x_{i,t+1}/y_{t+1}$. Thus, these ratios can take values between 0 and 1 and $\sum_{i=1}^{N} d_{i,t+1} = 1$.

As we do not know the current ratio at time $t+1$, we need to decide which and how many past ratios to use as well as

Final edited form was published in "IDEAS '12: 16th International Database Engineering & Applications Symposium. Prague 2012", S. 120–129, ISBN 978-1-4503-1234-9
https://doi.org/10.1145/2351476.2351490

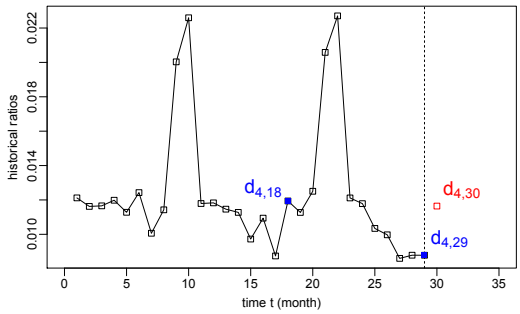


Figure 3: Calculation of Inclusion Probabilities.

on an aggregation scheme. We can make two observations. If the ratios highly fluctuate over time, we can choose a ratio close to the current time $t+1$ that might resemble the next ratio best. If the ratios are approximately stable, the type of ratio probably has quite a low influence, so we can choose a ratio close to the current time $t+1$ as well.

However, in addition, it might make sense to choose a ratio according to the seasonality of the data (see example in Figure 3). We again use a base time series $i = 4$ from our sales real-world data set, containing the total sales for audio devices of retailers in Czech Republic according to different months. Figure 3 displays the historical ratios $d_{i,t}$ for different months over time. Assume we want to forecast the 30th month and the time series $i = 4$ is part of our sample. Therefore, we need the ratio $d_{4,30}$ as inclusion probability. It probably makes sense to use one of the recent ratios, for example the last ratio $d_{4,29}$. In addition, we can see a strong seasonal effect in the ratios, which is related to the seasonal effects in the time series itself. This is an annual effect as people tend to show different shopping habits at different times of the year (e.g., Christmas). Therefore, it might also make sense to use the ratio 12 month before $d_{4,(30-12)} = d_{4,18}$.

To conclude, we propose a combined strategy, where we use a mixture of past ratios close to the current point in time and the ratio one season ago:

$$\hat{d}_{i,t+1} = \alpha \cdot \frac{\sum_{x=0}^{k-1} d_{i,t-x}}{k} + (1-\alpha) \cdot d_{i,t+1-season}. \quad (4)$$

We therefore average over the last k ratios and include the ratio one season ago. The seasonality is usually the same for the whole data set and known by the forecast models. If it is not available, we just average over the last k ratios. The higher the number of past values k, the more robust we are to outliers. However, we might loose in accuracy. The parameter α determines the weight of the seasonal ratio. The optimal setting of these parameters depends on the characteristics of the time series data. We tried different parameters for k and α using the data sets from the experimental evaluation (see Subsection 6.1). We observed that a value of $k = 3$ (using the last three ratios) and $\alpha = 0.5$ (uniform weighting) leads to a good accuracy in most cases.

The final inclusion probabilities are calculated by summing over all probabilities within the sample and normalizing with the sum over all probabilities ($= 1$). As a result, the aggregate forecast $\hat{y}_{t+1}$ at time $t+1$ is estimated by

$$\hat{y}_{t+1} = \frac{1}{\sum_{X_i \in S} \hat{d}_{i,t+1}} \cdot \sum_{X_i \in S} \hat{x}_{i,t+1}, \quad (5)$$

where the $\hat{d}_{i,t+1}$ are estimated using Formula 4.

Our proposed estimator can be used ad-hoc to answer an aggregation query from an available sample of base models as it does not require any parameter estimation.

4. REFINE

If some real data becomes available, this data should replace the base forecasts to obtain a better aggregate forecast. In this section, we discuss different possibilities to converge towards the real aggregate forecast values as fast as possible. We first introduce three basic approaches and there parameters in Subsection 4.1. Then, we discuss two approaches to set those parameters (Subsection 4.2 and 4.3).

4.1 Refine Basics

Let $R = \{x_{i,t+1}\}$ be the available real values of size m at time t, with $1 \leq i \leq m$. In simplest case, we can calculate the forecast $\hat{y}_{t+1}$ by scaling the sum over these real values:

$$\text{Refine I: } \hat{y}_{t+1} = \alpha \cdot \sum_{X_i \in R} x_{i,t+1}. \quad (6)$$

This first naïve approach only scales the real values and does not include any forecast values. However, if we use an appropriate scaling α, this simple method might already lead to good results.

Subsequently, the second approach additionally includes the available sample of forecasts $S = \{\hat{x}_{i,t+1}\}$. Let $\overline{S}$ be the remaining sample of forecasts $\overline{S} = S \setminus R$ of size o (i.e., forecasts where no real value is available). We can calculate $\hat{y}_{t+1}$ by using real values if available and forecasts otherwise:

$$\text{Refine II: } \hat{y}_{t+1} = \alpha \cdot \left(\sum_{X_i \in R} x_{i,t+1} + \sum_{X_i \in \overline{S}} \hat{x}_{i,t+1} \right). \quad (7)$$

The superiority of one method over the other strongly depends on the quality of the forecasts. If we have forecasts with a low accuracy, refinement with just the real values might work better. However if we have reasonable forecasts, the inclusion of those forecasts strongly improves the aggregate forecast. In order to combine both approaches into one method, we can examine the error of the available real values and include this error into our calculation. The key idea is to use the real values to evaluate the predictions and correct the remaining predictions using an estimate of the overall error:

$$\text{Refine III: } \hat{y}_{t+1} = \alpha \cdot \left(\sum_{X_i \in R} x_{i,t+1} + \sum_{X_i \in \overline{S}} \hat{x}_{i,t+1} \right) - \beta \cdot \sum_{X_i \in R} e_{i,t+1}. \quad (8)$$

We subtract the sum over the single errors $e_{i,t+1}$ scaled by some factor β from our estimated forecast. We therefore view the errors for the gathered real values as sample of the set of all prediction errors. If no significant drift occurred in the data and hence the models are reasonable accurate, the expected error will become zero. Otherwise the overall prediction is corrected by the probable systematic drift.

In the following, we discuss how to set these scaling parameters and how to calculate the errors.

Final edited form was published in "IDEAS '12: 16th International Database Engineering & Applications Symposium. Prague 2012", S. 120–129, ISBN 978-1-4503-1234-9
https://doi.org/10.1145/2351476.2351490

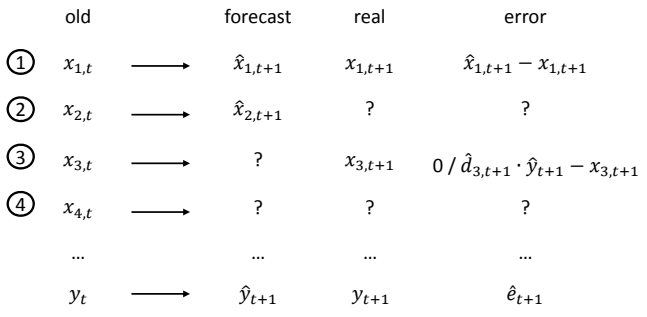


Figure 4: Refine III Overview

4.2 Uniform Refine

If we do not have any other information besides the forecasts and real values, we can only perform uniform scaling. Therefore, the parameter α is calculated by

$$\text{Refine I: } \alpha = \frac{N}{m} \tag{9}$$

$$\text{Refine II/III: } \alpha = \frac{N}{m+o}, \tag{10}$$

where N are the total number of time series, m is the number of available real values and o is the number of remaining base forecast values in the sample.

For refine III, we need to determine the errors of the available real and forecast values. We can distinguish four different cases (Figure 4). If we have no real value (case 2 and 4), we can not calculate any error, regardless of the existence of a forecast values. If we do have a real value as well as a forecast value (case 1), we can calculate the true model error by the taking the difference between those two:

$$e_{i,t+1} = \hat{x}_{i,t+1} - x_{i,t+1}. \tag{11}$$

If we do have a real value but no forecast value, we can not calculate a true error (case 3). We therefore assume an error of 0 for uniform refine. We extend this in Subsection 4.3.

In order to scale the errors, we use again uniform scaling. However, we need to account for the number of included real values, since real values replace estimates and therefore do not contribute to the overall error anymore. Therefore, the errors are scaled as follows:

$$\beta = \frac{N-(n-o)}{(n-o)}, \tag{12}$$

where $n-o$ is the number of determined errors (sample size n minus remaining sample o).

4.3 Refine with Historical Ratios

Similar to our aggregation approach, we can improve the refinement of forecast values (scaling as well as error calculation) if we use the information in the history of the time series:

$$\text{Refine I: } \alpha = \frac{1}{\sum_{X_i \in R} \hat{d}_{i,t+1}} \tag{13}$$

$$\text{Refine II/III: } \alpha = \frac{1}{\sum_{X_i \in R \cup \overline{S}} \hat{d}_{i,t+1}} \tag{14}$$

The estimator now requires the weights of the real values for refine I and the weights of the real as well as the forecast values for refine II and III.

For refine III, we extend the error calculation for case 3 of Figure 4. As we know the estimated aggregate forecast $\hat{y}_{t+1}$ and the historical ratios $\hat{d}_{i,t+1}$, we can estimate the forecast value of the base time series $\hat{x}_{i,t+1}$ by scaling down the estimated aggregate forecast:

$$\hat{e}_{i,t+1} = \hat{d}_{i,t+1} \cdot \hat{y}_{t+1} - x_{i,t+1}. \tag{15}$$

As more and more real values arrive, the accuracy of the aggregate forecast $\hat{y}_{t+1}$ increases and thus we can retrieve a better estimate of the base forecast value $\hat{x}_{i,t+1}$.

We again propose to the use the historical ratios in order to scale the errors:

$$\beta = \frac{1 - \sum_{X_i \in R} \hat{d}_{i,t+1}}{\sum_{X_i \in R} \hat{d}_{i,t+1}}. \tag{16}$$

In the bottom part, we include the historical ratios of all available real values as we calculate an error over all of them. We again do not include those values to estimate the *overall error* and therefore subtract the weights from the sum over all weights.

5. CONFIGURATION

In the last two sections, we discussed how to estimate and refine an aggregate forecast from an available sample of base forecasts. In this section, we shortly discuss how to actually design such as sample.

Consider N time series, our goal is to choose a sample $S = \{X_i\}$ of base time series for a given sample size n that minimizes the error of the one-step ahead aggregate forecast:

$$\min_S \text{error}(\hat{y}_{t+1}, y_{t+1}). \tag{17}$$

The aggregate forecast $\hat{y}_{t+1}$ is estimated by Equation 5.

In the naïve case, we could apply uniform sampling and choose the sample randomly. However, uniform sampling has the well known disadvantage of ignoring the variance in the data distribution [8]. We can achieve a higher accuracy if we use a weighted sampling scheme. In weighted sampling designs, the probability of an item included into the sample varies among the items in the population [19]. Similar to aggregation, the challenge is to determine reliable weights in order to achieve a high accuracy, where we distinguish two different cases.

First, we do not have built forecast models for the time series $X_1 ... X_N$. In this case, we can only use the information in the history of the time series. Similar to our aggregation approach, we need to choose those time series with a higher probability that contribute the most to the overall aggregate forecast. However, in contrast to our aggregation approach we do not want to maximize the accuracy for just the next aggregate forecast but we want to choose a sample that maximizes the accuracy of possible all future aggregate forecasts. We therefore average over all past ratios and the weights are calculated by

$$w_{i,t} = \frac{1}{t} \sum_{j=1}^{t} d_{i,t-j}. \tag{18}$$

Second, we already have built base models M_i for all base time series X_i. These models were built for evaluation purpose and our goal is to choose only a subset of those models to store and, more importantly, maintain in the database. Here, we can additionally use the information available in

Final edited form was published in "IDEAS '12: 16th International Database Engineering & Applications Symposium. Prague 2012", S. 120–129, ISBN 978-1-4503-1234-9
https://doi.org/10.1145/2351476.2351490

the models. The main assumption is that it would be wise to choose time series that have a low model error and therefore produce good forecasts. For this we use the in-sample error of the models, i.e., the error is calculated over the training data by using the best parameter combination of the model. The final weights for weighted sampling are given by

$$w_{i,t} = \frac{1}{\text{in\_sample\_error}(M_i)} \tag{19}$$

and normalized with the sum over all in-sample model errors.

Note, in both cases, we could also sort the time series by their weights in descending order and choose the top-n time series. This is a special case of our general weighted sampling approach.

If the weights change, either the contribution of the time series to the aggregate series or the accuracy of the models, we need to adapt our sample. A simple approach would be to periodically check the weights of the time series and recalculate the sample if a significant change occurred.

6. EXPERIMENTAL EVALUATION

We conducted an experimental study to evaluate the (1) overall performance (accuracy and speedup) of our approaches on four real-world data sets as well as the accuracy and characteristics of our (2) aggregation, (3) refine and (4) configuration estimators.

6.1 Experimental Setting

To implemented our approaches, we used the statistical computing software environment R (`http://www.r-project.org/`). It provides efficient build-in forecast methods and parameter estimation approaches, which we used to build the forecast models. All experiments were executed on an IBM Blade with two processors (each a Dual Core Intel Xeon at 2.80 GHz) and 4 GB RAM.

We used one synthetic data set and four different real-world data sets:

- *Synthetic data:* The synthetic data set is randomly generated and serves the purpose of showing the general characteristics of the different approaches. We did not generate complete time series. We only randomly generated the real values of base time series $X_1 ... X_N$ (between 0 and 1) at time $t+1$. The aggregate value y_{t+1} is then the sum of those random values. We generated $1,000$ values, so, for uniform probability distribution functions, the expected aggregate value is about 500. Then, we created three data sets of forecast values for time $t+1$. The first data set *no error* assumes that we could create perfect models and the forecast value are the same like the real values at time $t+1$. The second data set *random* adds a random error of maximum 20% to the forecasts. The final data set *systematic* adds a positive error of 10% to the forecast values, thus simulating a systematic error.

- *Retailer sales:* The first real-world data set contains 32 time series of monthly sales (February 2004 to May 2009) of audio devices according to different countries.

- *Wind energy:* For the third data set, we use the publicly available NREL wind integration data set [3]. We choose about 80 time series at random from this data set, which contain supply of wind energy for 2004 to 2006 in a resolution of one day.

- *Worldwide electricity generation:* This data set includes metered world-wide electricity generation from the US Energy Information Administration [2]. It consists of 1344 time series according to different countries, states and products from the years 1980 to 2008 in an annual resolution.

- *Australian domestic tourism:* The final data set consists of observations on the number of visitor nights for the Australian domestic tourism [1]. The number of visitor nights were recorded according to different states and purposes of visit for the years 2004 to 2010 in a quarterly resolution.

To build the base and aggregate models for the real-world data sets, we use exponential smoothing models [22] as well as autoregressive integrated moving average (ARIMA) models [7]. Both are well examined [17], have shown empirically to be able to model a wide range of real world time series [23] and are usually computationally more efficient than elaborate machine learning approaches. In order to measure the accuracy of our approaches, we use the symmetric mean absolute percentage error (SMAPE), which is a scale-independent accuracy measure and takes values between 0 and 1, where 0 refers to perfect accuracy. It is defined as:

$$SMAPE = \frac{|x_t - \hat{x}_t|}{x_t + \hat{x}_t}, \tag{20}$$

where x_t describes the real value of a time series at time t and $\hat{x}_t$ the corresponding forecast value.

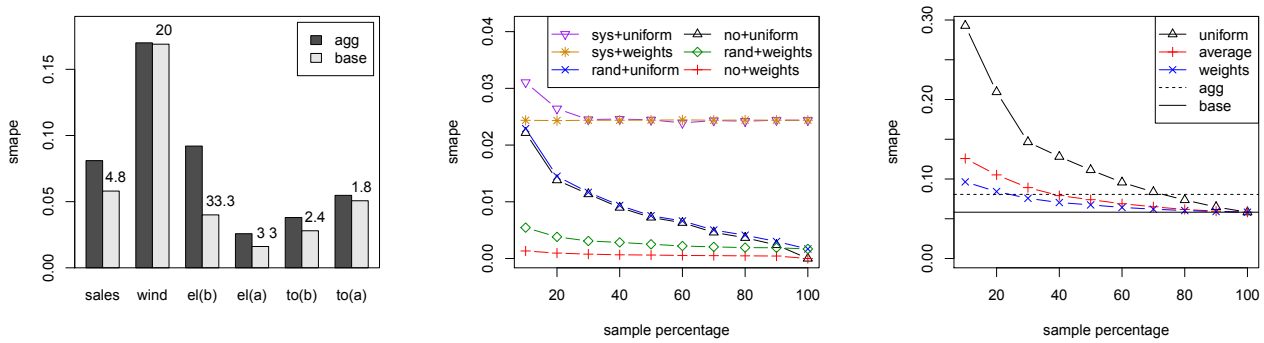
We used about 50% of the values of all time series to train the models and the remaining 50% to evaluate our approaches. We always updated the models and historical ratios after each new real time series value. We averaged over $1,000$ runs for all experiments.

6.2 Overview

To show the general benefit of our approach, we provide first an overview over all data sets. Then, in the following subsections, we discuss the characteristics of our approach on selected data sets. In Figure 5(a) the forecast error of all data sets using an aggregate model (*agg*) and using all base models (*base*) is shown. We observe that we can reach a better accuracy by aggregation of base forecasts for almost all data sets. Only the wind data set (*wind*) shows hardly a difference of the two approaches. The reason is that wind data is hard to predict and we are not able to build a very accurate model on aggregate as well as on base level. For the electricity (*el*) and tourism data set (*to*), we have several aggregation possibilities as the time series are described by more than one attribute. We, therefore, provide the best aggregation query with the highest error difference of base models to an aggregate model (*el(b)* and *to(b)*) and the average error (*electr(a)* and *tour(a)*) over all aggregation queries.

On top of the errors in Figure 5(a), we noted the speedup in terms of model maintenance costs of our sampling approach over using all base models. This speedup is calculated by using our best sampling approach (weighted sampling with historical ratios) and by using the sample size that achieves the same accuracy as all base models. We permitted a tolerance of 5%. For all data sets, we achieve at least

Final edited form was published in "IDEAS '12: 16th International Database Engineering & Applications Symposium. Prague 2012", S. 120–129, ISBN 978-1-4503-1234-9
https://doi.org/10.1145/2351476.2351490



(a) Overview Accuracy and Speedup (b) Aggregation - Synthetic Data (c) Aggregation - Sales Data

Figure 5: Overview and Aggregation Experimental Results.

a speedup of 2. Thus, in general, we need to maintain just half of the forecast models (sample percentage = 50%) to reach the same accuracy as using all base models. The sales and electricity data set perform best in terms of speedup (4.8 and 33.3 respectively) and accuracy improvement.

6.3 Aggregation

In a second series of experiments, we measured the accuracy of our aggregation estimators with increasing sample size. In Figure 5(b), the SMAPE for the synthetic data set is shown, using our uniform as well as our adapted estimator (Equation 5). To calculate the historical ratios for the synthetic data set, we use the actual ratio of the real values with a random error of 5%.

For the forecasts with no error (*no+weight*), we converge fast towards an error of zero using our adapted estimator. In contrast, the uniform estimator leads to a much worse error and slowly converges towards an error of 0 with increasing sample size (*no+uniform*).

For the forecasts with random error, we can only reach a certain minimum error using all forecasts. Both, the uniform and adapted estimator converge towards this error. The uniform estimator is similar to the estimator of the forecasts with no error as the random forecast errors compensate each other out (*rand+uniform*). In contrast, our adapted estimator shows a much better accuracy (*rand+weights*).

For the forecasts with systematic error, the adapted estimator shows a constant error (*sys+weights*). This is the best possible error as all forecasts have a similar systematic error. In contrast, the uniform estimator starts worse and converges towards this minimum error (*sys+uniform*).

We performed the same experiment using the sales data set (Figure 5(c)). The straight lines shows the error when aggregating over all *base* forecasts while the dotted line shows the error when creating a model for the aggregate time series (*agg*). We can see that the aggregation of the base forecasts leads to a higher accuracy than the forecast over the aggregate model. In addition, we measured the accuracy using the uniform estimator (*uniform*) and our adapted estimator (*weights*) with the combined inclusion probabilities (Equation 4). The adapted estimator leads to a high accuracy even with small sample sizes. With a sample size of 30% the accuracy is better than the accuracy of a model over the aggregate series. In order to show the benefit of our suggested inclusion probabilitiy approach, we also plotted the accuracy of using our adapted estimator with an average over all historical ratios (*average*) as inclusion probability. Our combined strategy achieves a maximum improvement of about 40% compared to using all ratios. We noticed the same characteristics of the different approaches for the other data sets and thus omit detailed figures.

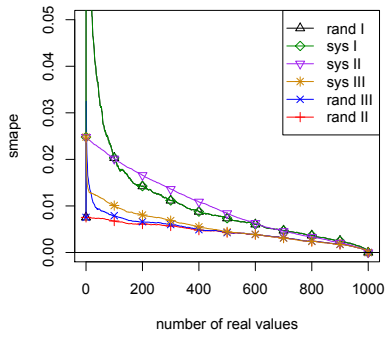
6.4 Refine

In this subsection, we experimentally evaluate the accuracy of our refinement approach according to different number of real values available and to different sample sizes.

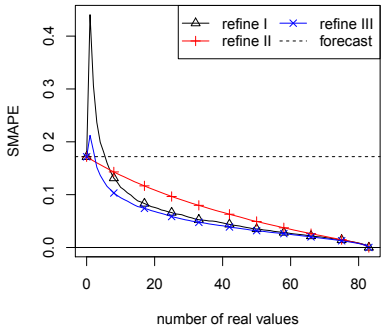
We start with uniform refine and the synthetic data set, leaving out the forecasts with no error. Figure 6(a) shows the accuracy for a 50% sample of forecast values with increasing number of real values included in the aggregate forecast. We can see that refine I is the same for the forecasts with random (*rand I*) as well as the forecasts with systematic error (*sys I*) as it only includes the real values. It shows a high outburst in the beginning and then converges towards an error of zero. Refine II shows a higher accuracy for the random errors (*rand II*). For the systematic errors (*sys II*), it is better than refine I in the beginning but similar in the end as the forecasts have a strong bias. Refine III captures this error for the synthetic data set (*sys III*) and shows a much better accuracy than the other two approaches. For the random error (*rand III*), we can not filter out such an error but nevertheless refine III shows only an outburst in the beginning and than converges towards refine II as the errors cancel each other out.

Figure 6(b) displays the accuracy of the wind data set. Similar to our synthetic data set with systematic errors we can see a strong outburst of refine I in the beginning. However, it then shows a better accuracy than refine II. Again, refine III shows the best accuracy with only a small outburst in the beginning as there are not enough real values available to make accurate error estimates. Figure 6(c) shows the experiment for a fixed number of real values (50% of all real values) and increasing sample sizes. We can see that refine I stays approximately constant as it is independent from the sample size. Refine II gets worse with increasing sample size as more forecasts with a high error are included in the

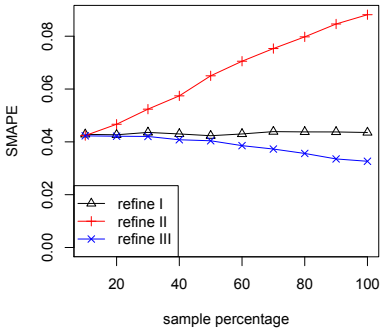
Final edited form was published in "IDEAS '12: 16th International Database Engineering & Applications Symposium. Prague 2012", S. 120–129, ISBN 978-1-4503-1234-9
https://doi.org/10.1145/2351476.2351490



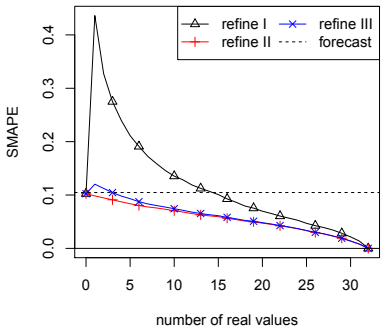
(a) Synthetic Data - 50% Sample



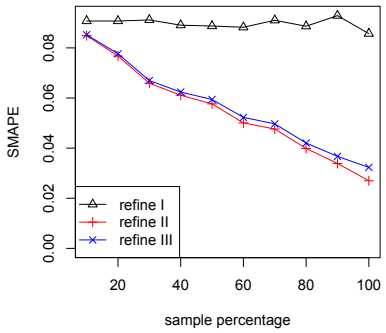
(b) Wind Data - 50% Sample



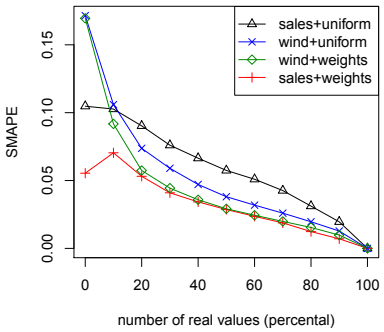
(c) Wind Data - 50% Real Data



(d) Sales Data - 50% Sample



(e) Sales Data - 50% Real Data



(f) Uniform vs. Historical Ratios - 50% Sample

Figure 6: Refine Experimental Results.

aggregate forecast. In contrast, refine III improves with a higher sample size as we can retrieve a better estimate of the forecast error.

We repeated the same experiment for the sales data set. In Figure 6(d) the development of the forecast accuracy for a 50% sample with increasing number of real values is shown, while Figure 6(e) shows the development of the forecast accuracy with increasing sample size if 50% of the real data is available. Here, refine I shows a very bad forecast accuracy, while refine II and III behave very similar. For this data set, our forecasts are quite accurate. Therefore, we can not compensate any forecast error and refine III is even slightly worse with a low number of real values. With increasing sample size, refine II and III strongly improve as we can include more base forecast values to calculate the aggregate forecast.

To conclude, with refine III we can capture a probably systematic error and result in a higher accuracy than refine I and refine II. If the forecasts are quite accurate, we retrieve a similar accuracy than refine II.

Refine with historical ratios shows the same characteristics as uniform refine in general. Therefore, in a last experiment we only compare uniform refine and refine with historical ratios (Figure 6(f)). We display the forecast accuracy of both data sets (wind and sales) for the two approaches (uniform and weights) with increasing number of real values using a 50% sample of forecast values. For both data sets, refine with historical ratios shows a better accuracy and convergence than uniform refine. We can see a higher improvement for the sales data set than the wind data set as all base time series of the wind data set have a similar contribution to the aggregate series.

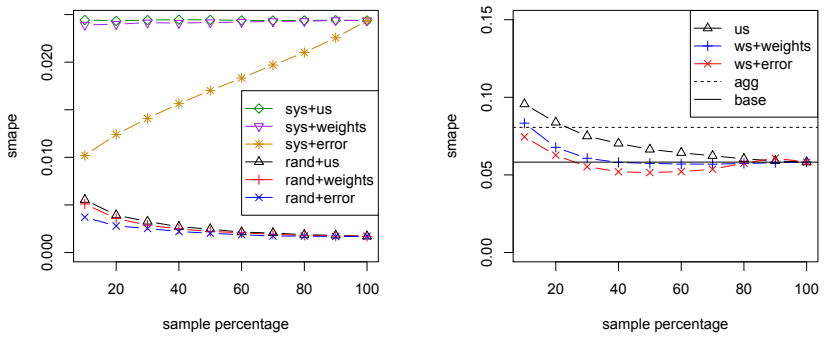
6.5 Configuration

In this section, we compare the best uniform sampling from Subsection 6.3 with weighted sampling. For weighted sampling, we used the historical ratios as weights (Equation 18) as well as the in-sample model error (Equation 19). We again measured the SMAPE of the approaches for different sample sizes.

We first show the results of the synthetic data set (Figure 7(a)). For the forecast with random error, we can only see a small improvement of weighted sampling (*rand+weights* and *rand+error*) over uniform sampling (*rand+us*). Due to a uniform distribution of the historical ratios, our uniform sampling approach already performs quite well and we only yield a small improvement using weighted sampling with historical ratios. As the model errors are also uniform distributed, the weighted sampling approach using the errors itself leads to a small improvement as well.

For the forecasts with systematic error we can see that

Final edited form was published in "IDEAS '12: 16th International Database Engineering & Applications Symposium. Prague 2012", S. 120–129, ISBN 978-1-4503-1234-9
https://doi.org/10.1145/2351476.2351490



(a) Synthetic Data (b) Sales Data

Figure 7: Configuration Experimental Results

uniform (*sys+us*) and weighted sampling with historical ratios (*sys+weights*) perform equally. Both sampling schemes already reach the minimum possible error with small sample sizes as the systematic errors are distributed equally. The error of the weighted sampling approach using the errors as weights (*sys+errors*) increases with increasing sample size. The reason is that all forecasts have a systematic error. If we first choose the forecasts with a small error and estimate the total aggregate from those, we actually propagate a smaller error to the total aggregate and hence reach a better accuracy with a smaller sample size.

The experimental results for the sales data set is shown in Figure 7(b). Our weighted sampling scheme with the historical ratios (*ws+weights*) leads to good improvement over the uniform sampling approach (*us*). With a sample size higher than 10% we reach a better accuracy than a model over the aggregate time series. The weighted sampling scheme with the in-sample errors as weights(*ws+errors*) further improves the result. Similar to our synthetic data set, for certain sample sizes, we can even see a better accuracy with a sample of base models compared to using all base models.

7. RELATED WORK

Related work can be found in three main areas: (1) approximate query processing, (2) existing approaches to integrate forecasting in database management systems and (3) hierarchical forecasting studies in forecasting literature.

Approximate Query Processing Sampling based methods have been used in a wide variety of scenarios in databases, such as query selectivity estimation in query optimization, providing sampling as a relational operation, and approximate query processing [24]. Our work is probably most related to the area of approximate query processing. Current research focuses on online techniques [20] as well offline techniques [8, 15]. For example, Ganti et. al. have developed a weighted sampling schemes that exploits workload information (i.e., query frequency) to continuously tune a representative sample of the data [15]. However, our work differs in two main points. First, we want to approximately process forecast queries that already provide approximate answers. Thus, with a sample of forecast values we can still provide the same accuracy as using all forecast values. Second, we deal with the approximate aggregation of time series values, while current research focuses on single values. This gives us the unique possibility of including the history of the time series to improve the estimate.

Forecasting in DBMS Recent research has addressed the integration of time series forecasting inside a DBMS. Within the Fa system [10] an incremental approach is proposed to build models for a multidimensional time series in which more attributes are added to the model in successive iterations. Furthermore, the skip-list approach for efficient forecast query processing [16] proposes an I/O-conscious skip list data structure for very large time series in order to enable the determination of a suitable history length for model building. However, all approaches investigate how to efficiently find the best forecast model for one specific forecast query using database techniques. They do not address the aggregation of models or the refinement of forecasts if new values arrive asynchronously. Agarwal et al. address the problem of forecasting high-dimensional data over trillions of attribute combinations [4]. They propose to store and forecast only a sub-set of attribute combinations and compute other combinations from those using high-dimensional attribute correlation models. In contrast to our approach, they calculate several forecasts from one base model using disaggregation, while we calculate one forecast from several base models using sampling to provide a higher accuracy.

Hierarchical Forecasting The aggregation challenge was also examined in specific forecasting and economic literature [14, 11, 26]. Different studies examined the performance of bottom-up approaches (base time series forecasts are made directly and aggregated to get higher level forecasts) and top-down approaches (forecasts are made at aggregate level and then allocated to lower levels). Influencing factors of the superiority of one approach over the other were investigated, e.g., quality of forecast method, correlation between variables and forecast errors [6]. However, all studies only consider complete aggregation and do not address the possibility of aggregation over time series samples.

Gross and Sohl published 21 disaggregation methods [18] that can be used to scale down an aggregate forecast in order to retrieve a base time series forecast. They analyzed simple approaches that use the last available value or the average over the whole historical time series as well as more complex approaches that additionally consider correlation

Final edited form was published in "IDEAS '12: 16th International Database Engineering & Applications Symposium. Prague 2012", S. 120–129, ISBN 978-1-4503-1234-9
https://doi.org/10.1145/2351476.2351490

of past values. They came to the conclusion that simple approaches work best. We reused the idea of disaggregation weights in order to obtain better estimates of the aggregate forecast. For this, we proposed a new combined method of that also uses information about the seasonality of the data.

8. CONCLUSIONS

The integration of forecasting inside a DBMS is a rising topic in the research community. In this paper, we addressed the optimization of aggregation forecast queries. We discussed how to calculate an aggregate forecast from just a sample of base forecasts and how to refine the aggregate forecast if new real values arrive asynchronously. In our experimental evaluation, we have shown that with a small sample of base forecasts, we can reach a higher accuracy than a forecast provided by an aggregate model. As a data warehouse usually contains a high number of aggregation possibilities, we can save aggregate models and use an available sample of base models to answer ad-hoc aggregation queries without loosing any or only a little accuracy.

The proposed approach is in its initial version so that there is still room for improvement. First, the current approach can be extended to support other aggregation functions (e.g., maximum, variance, quantiles). Different estimators have to be developed that calculate and refine the aggregate forecast value. Secondly, we began to discuss how to design a sample of base time series to increase the accuracy of an aggregate forecast. This might be extended to multiple aggregation hierarchies, where one base time series can contribute to several aggregate time series. Finally, we want to explore how our results can be applied to the area of online aggregation, where an initial result is provided very fast and then successively refined over time.

Acknowledgment

The work presented in this paper has been carried out in the MIRABEL project funded by the EU under the grant agreement number 248195.

9. REFERENCES

[1] *Tourism Research Australia - National Visitor Survey*, 2012. `http://www.ret.gov.au/tourism/research/tra/Pages/default.aspx`.

[2] *US EIA - International Energy Statistics*, 2012. `http://tonto.eia.doe.gov/cfapps/ipdbproject/IEDIndex3.cfm?tid=2&pid=2&aid=2`.

[3] *Wind Integration Datasets*, 2012. `http://www.nrel.gov/wind/integrationdatasets/`.

[4] D. Agarwal, D. Chen, L. Lin, J. Shanmugasundaram, and E. Vee. Forecasting high-dimensional data. In *SIGMOD*, 2010.

[5] M. Akdere, U. Çetintemel, M. Riondato, E. Upfal, and S. Zdonik. The case for predictive database systems: Opportunities and challenges. In *CIDR*, 2011.

[6] A. Barnea. An analysis of the usefulness of disaggregated accounting data for forecasts of corporate performance. *Decision Sciences*, 11:17–26, 1980.

[7] G. Box, G. Jenkins, and G. Reinsel. *Time Series Analysis: Forecasting and Control*. Wiley, 2008.

[8] S. Chaudhuri, G. Das, and V. Narasayya. Optimized stratified sampling for approximate query processing. In *TODS*, 2007.

[9] J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein, and C. Welton. Mad skills: New analysis practices for big data. In *VLDB*, 2009.

[10] S. Duan and S. Babu. Processing forecasting queries. In *VLDB*, 2007.

[11] D.M. Dunn, W.H. Williams, and T.L. DeChaine. Aggregate versus subaggregate models in local area forecasting. *Journal of the American Statistical Association*, 71:68–71, 1976.

[12] U. Fischer, M. Böhm, and W. Lehner. Offline design tuning for hierarchies of forecast models. In *BTW*, 2011.

[13] U. Fischer, F. Rosenthal, and W. Lehner. F2db: The flash-forward database system (demo). In *ICDE*, 2012.

[14] G. Fliedner. Hierarichal forecasting issues and use guidelines. *Industrial Management & Data Systems*, 101:5–12, 2001.

[15] V. Ganti, M. L. Lee, and R. Ramakrishnan. Icicles: Self-tuning samples for approximate query answering. In *VLDB*, 2000.

[16] T. Ge and S. Zdonik. A skip-list approach for efficiently processing forecasting queries. *VLDB*, 2008.

[17] J. G. D. Gooijer and R. J. Hyndman. 25 years of time series forecasting. *International Journal of Forecasting*, 22:443–473, 2006.

[18] C. W. Gross and J. E. Sohl. Disaggregation methods to expedite product line forecasting. *Journal of Forecasting*, 9:233–254, 1990.

[19] M. Hanif and K. Brewer. Sampling with unequal probabilities without replacement: A review. *International Statistical Review*, 48:317–335, 1980.

[20] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. 1997.

[21] D. Horvitz and D. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260):663–685, 1952.

[22] R. J. Hyndman, A. B. Koehler, R. D. Snyder, and S. Grose. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting*, 18, 2000.

[23] S. Makridakis and M. Hibon. The M3-Competition: results, conclusions and implications. *International Journal of Forecasting*, 16:451 – 476, 2000.

[24] F. Olken. Random sampling from databases, 1993.

[25] F. Rosenthal and W. Lehner. Efficient in-database maintenance of arima models. In *SSDBM*, 2011.

[26] A. Zellner and J. Tobias. A note on aggregation, disaggregation and forecasting performance. *Journal of Forecasting*, 19:457–469, 2000.