# Database support for automotive analysis *

Dennis Marten, Holger Meyer, and Andreas Heuer[0000−0002−6163−6649]

Institute of Computer Science, Rostock University, Albert-Einstein-Strasse 22, 18059 Rostock, Germany { dm, hme, ah }@informatik.uni-rostock.de

**Abstract.** Based on an analysis of typical automotive measurements data as found in ASAM MDF files, we derive requirements managing these data in a database system and create a mapping to a relational database structure. The performance of a parallel relational database solution is compared with a Python-based direct access and querying time-series in Python and with big data frameworks such as Apache Spark in different scenarios. A hybrid approach using some object-relational features of PostgreSQL performs best in most cases.

**Keywords:** automotive data · time-series data · time merge · relational database systems · ASAM MDF files

## 1 Motivation and Problem Specification

In the automotive industry, due to the continuous growth of collected sensor data, traditional hardware and software setups come to their limits. Within a joint project with an industrial partner, we developed and evaluated database oriented solutions to offer scalable automotive analysis.

The basis for this project is the **Measurement Data Format** (MDF). **MDF** is a binary file format that has been originally developed for automotive measurement and analysis in the 1990's. It has been officially standardized with version 4.0 (MDF4) by the Association for Standardization of Automation and Measuring Systems (**ASAM**) in 2009 and is even today the standard for storing and reading automotive sensor data in industrial usage [1]. While MDF4 files do not have any restriction on its actual size, analyzing a set of large or even many moderate sized files can become hardware demanding for local setups. Neglecting possible main memory shortages on long measurements, the disk storage problem might be worked around using network drives. Due to unfiltered data communication, this approach has shown to perform poorly even on comparatively small amounts of data.

In order to overcome these data size limitations, we started a cooperation project with an industrial partner and examined solutions that are based on database support with Python front ends using a conventional vertical architecture [7]. The main goal of this project was to survey a variety of database

systems on automotive analysis for industrial usage using a variety of different settings. Hereby, **two groups of systems** have been taken into consideration:

- (Parallel) relational database systems and
- Apache Spark with either NoSQL database systems or HDFS as data sources.

Besides the factor of scalability, relational database approaches generally promise several additional advantages, like fast selective queries via index structures, physical and logical optimization, data security aspects, as well as easy integration in current IT-setups. On the other hand, Apache Spark has shown to be a promising parallelization framework for big data applications. Both groups should have been evaluated using three given methods of frequently used basic automotive analysis tasks, while varying between **three different storage and computation schemes**:

1. storing data locally or on a cluster setup in order to allow analysts using the same API for different purposes,
2. storing converted floating-point data or raw fixed-point data with the respective conversion rules and parameters for online conversion, and
3. pushing different sub-methods into the database system (arithmetic operations and time merges, i.e. time series interpolation) in order to decrease communication costs and the need of resource demanding local computations.

The **three tasks of automotive analysis** we evaluated can be categorized into two groups:

- The first, more data intensive group, is selecting time series ("channels") of a number of files (measurements). All channels of one measurement are interpolated with respect to one of the time axes, so all channels ultimately share one time axis, allowing for pointwise comparison or calculation. The interpolated data is either visualized, manipulated (for instance using arithmetic operations) or analyzed, e.g. finding time intervals where channel values are in certain ranges.
- The second group can be refered to as meta-information queries: the goal is to find measurements under constraints, like a concrete date of recording or some specific time axis information.

Due to space limitations, we would like to refer to an extended technical report [7] for a more detailed presentation of the problem specification and a State of the Art analysis of time series management in database systems.

In the following section, we give a brief overview of the course of the presented project.

## 2 Course of Project

We started our investigation focusing on relational database systems, as these seemed to be best suited for (1) the comparatively small amount of test data

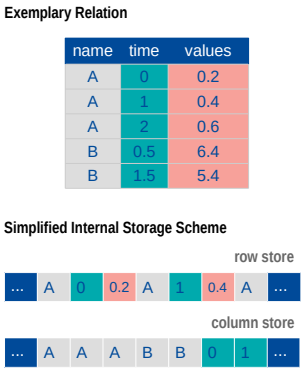


**Fig. 1.** 1NF relational scheme.

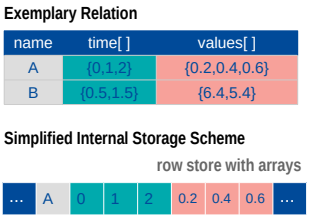


**Fig. 2.** Object-relational scheme.

(around 40 GB) that was available at the start of the project and (2) the evaluation of quite selective algorithms.

After evaluating first drafts of database schemas, we found that providing a schema that enables systems to transparently apply compression techniques is a key factor for our automotive scenario. Therefore, we distinguished **three different setups**:

- row stores,
- column stores,
- and row stores that allow the use of array data types (such as object-relational database systems).

Since automotive measurement data are heavily compressible, especially in the context of run length encoding (RLE), column stores for first normal form (1NF) relations or row stores with array data types (object-relational scheme) perform best (see Figures 1 and 2 for exemplary relations in the three different setups). With this differentiation, we developed **four different relational schemas** in order to satisfy the requirement of either storing raw data (bit varying) and its conversion rules, or converted floating point data (double precision), while enabling sequential data storage and therefore transparent run-length encoding. As an example, the (ER) schema using arrays in a row store with floating point storage is depicted in Figure 2.

As requested by our industrial partner, we implemented the adjustment of MDF data and the subsequent import into database systems in Python. For this, we needed to evaluate different packages for reading MDF-files and database communication.

For the three automotive analysis tasks we have implemented several database supported approaches. We took special focus on evaluating the pushdown of as many operations as possible into database systems as we have discussed in our recent work on SQL-based scientific computing [6,5,4,3]. We compared pure SQL implementations and UDF-based ones.

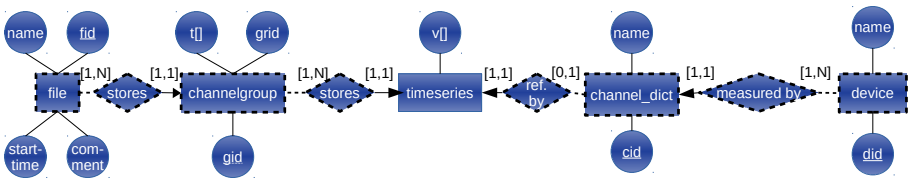


**Fig. 3.** Entity relationship model of the floating point schema for database systems that support arrays. Dashed borders in combination with arrows describe weak entity relationships.

After using RDBMSs, we evaluated **Apache Spark** with **different types of data storages** including the use of CSV- and Parquet-files in HDFS and requesting data from (parallel) RDBMSs and NoSQL-systems using (array-based) schemas similar to the ones for row stores with array support.

As a starting point, we reviewed different libraries for storing and analyzing timeseries in Apache Spark. Here, we have found that these have not been sufficient for automotive data [2]. Therefore, new implementations based on RDDs and DataFrames had to be developed and evaluated.

## 3 Results and Conclusions

We have found several noteworthy results that we can only summarize in this short paper. A more detailed description of all the results can be found in [7].

As a first conclusion, importing complete datasets of measurements into relational database systems via Python is ineffective. As the heavily compressed sensor data is decompressed in order to adjust its structure to the internal representation of the respective database system, the amount of data that is communicated is needlessly high. Solutions to this problem are either transforming sensor data in its actual compressed form before the communication process or fully integrating the import (if possible) into the database system as a UDF.

Regarding relational database schemas, we have found that row stores supporting array datatypes performed best. Here, he main reasons are superior compression and faster selections as relations contain so much lesser tuples, that even data selections via B-tree index structures become significantly faster. On the other hand, row stores that have to work on strictly first normal form relations are unusable for these kinds of operations as relation sizes inflate rapidly.

Evaluating the three aforementioned automotive methods has shown that the pushdown of interpolation (only applicable with UDFs) and arithmetic operations, as well as finding intervals under constraints are very effective in this scenario. In comparison to Python implementations on SSD drives for local calculation or using network drives for cluster computing, all of the three methods could experience a significant speed up when supported by (parallel) relational

database systems. Especially, inter-measurement queries for metainformation performed up to nearly 500 times faster, due to the very low communication cost and the selectivity of the problem.

The most promising relational database systems we have tested are PostgreSQL (local) and its parallel branch Postgres-XL (cluster setups). Besides superior performance, both systems run under BSD-like licenses and support a wide range of useful functionalities, like index structures, UDFs and array data types with transparent run length encoding. However, due to a relatively small amount of test data (40 GB) and fairly light-weight analysis, Postgres-XL could not benefit from its parallel computation capabilities, ultimately leading to a reduction of relative performance benefits obtained by database support in comparison to the local setup. Similarly, the initial overhead of Apache Spark has shown to be to overwhelming for the data sizes we evaluated, making a comparison to the relational systems unfair. Despite this, the most promising data storage for Apache Spark has shown to be Parquet files in HDFS.

As this project has only been meant as an initial survey of possible database support for automotive applications, there are many ways to continue the research, like evaluating larger data sets or more complex methods like pattern recognition (e.g., peak detection). Nonetheless, this project has shown that scalable solutions for automotive inter-measurement analysis can be done efficiently using relational database systems.

## References

1. Association for Standardization of Automation and Measuring Systems: ASAM MDF (2019), https://www.asam.net/standards/detail/mdf
2. Lutsch, A.: Effiziente Datenvorbereitung für Analysen im Automotive-Bereich. Bachelor Thesis, Rostock University (2019), http://eprints.dbis.informatik.uni-rostock.de/989/
3. Marten, D., Heuer, A.: A framework for self-managing database support and parallel computing for assistive systems. In: Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments, PETRA 2015, Corfu, Greece, July 1-3, 2015. pp. 25:1–25:4 (2015). https://doi.org/10.1145/2769493.2769526, https://doi.org/10.1145/2769493.2769526
4. Marten, D., Heuer, A.: Machine Learning on Large Databases: Transforming Hidden Markov Models to SQL Statements. Open Journal of Databases (OJDB) **4**(1), 22–42 (2017), https://www.ronpub.com/ojdb/OJDB_2017v4i1n02_Marten.html
5. Marten, D., Meyer, H., Dietrich, D., Heuer, A.: Sparse and Dense Linear Algebra for Machine Learning on Parallel-RDBMS Using SQL. OJBD **5**(1), 1–34 (2019), https://www.ronpub.com/ojbd/OJBD_2019v5i1n01_Marten.html
6. Marten, D., Meyer, H., Heuer, A.: Calculating Fourier transforms in SQL. In: Advances in Databases and Information Systems - 23nd European Conference, ADBIS 2019, Bled, Slovenia, September 8-11, 2019, Proceedings (2019)
7. Marten, D., Meyer, H., Heuer, A.: Database support for automotive analysis. Technical Report. Chair of Database and Information Systems, Rostock University, Rostock, Germany (September 2019), http://eprints.dbis.informatik.uni-rostock.de/995/