NoSQL database for air quality prediction

Petar Halachev
University of Chemical Technology and Metallurgy, Sofia, Bulgaria

Abstract: *The increasing application of NoSQL database technology and the neural networks raises the question of how compatible and applicable are the NoSQL databases to the neural network prediction models. This paper examines the applicability of a traditional relational database for storing air quality data and compares it to a NoSQL database performing the same functions. The possibility of the NoSQL database to feed a neural network model for predicting the atmospheric air quality is evaluated. The tendencies in the data are studied, and some solutions for improving the air quality are proposed. An analysis and a comparison of the performance of both relational SQL and NoSQL database systems by using real-world data for the Air Quality Index in the city of London is assessed and their performance is compared. Bivariate analysis on the data in order to assess the quality of the neural network forecast is performed.*
Keywords*: AIR QUALITY INDEX, NEURAL NETWORKS, NOSQL, PEARSON COEFICIENT*

1. Introduction

Recently, the document-based databases (NoSQL) are gaining more and more popularity. What they have in common is that the data is stored in the form of a table that is not fully normalized, and the level of normalization in the tables depends on the complexity of the data to be processed. Unlike relational databases (SQL based), which are organized around the concept for a table for each object in the information system. The various objects are connected to each other through relations (a relational tables for many-to-many relationships or index columns in the other cases). In the document [1] the applicability of the Cassandra NoSQL data base management system is considered. The latest releases of some NoSQL systems (Cassandra) use the so-called CQL (Cassandra Query Language), which resembles SQL in the way the queries are structured, as well as in the way the tables are defined (METADATA). For the most modern applications, Apache Cassandra is considered a good choice in the application and web services development environments.

It offers many advantages, including easy data portability on a global scale and outstanding performance. Today, Apache Cassandra is amongst the most popular NoSQL database management systems. It is used for online retail applications, for internet portals, for time series databases, for mobile applications, for internet banking.

The development of the right data model is critical to achieving a good application performance and future scalability. Creating the right data model is important for any application, but its importance increases when it comes to applications that require minimal latency, minimal blocking, and global scalability.

An application must be able to work both with thousands of records and several hundred concurrent users, and even when the number of records and users is in the millions or even billions figures. At the same time, the target is to minimize the waiting time.

Regardless of whether a SQL or NoSQL database management system is used, if the data model is not developed correctly or does not fit the underlying architecture of the chosen database, it can lead to performance degradation, long query wait times, data loss or corruption. It may be necessary to change the data model (migration) after the application is created and deployed, but this could prove to be an expensive and complex operation. Therefore, it is a good idea to select a proven methodology for designing a scalable, extensible and easy-to-maintain data model during the development process.

2. Differences between Cassandra and the relational databases

Most professionals are familiar with the relational databases such as Oracle, MySQL, SqLite, PostgreSQL. Before moving on to the data modeling with NoSQL, it is a good idea to pay attention to some differences between NoSQL-based and relational databases. Relational databases prioritize the data size minimization over the response time – a piece of information exists at only one place in the database (this is guaranteed by the Normalization rules, 1st, 2nd, 3rd, 4th Normal forms), but when retrieving data (SELECT statement) many joins (JOIN operations) have to be done. This results in slow execution of even trivial-to-fetch queries. NoSQL systems, on the other hand, prioritize the execution time for the retrieval query, and for this purpose, a given information is presented in several places in the database, and they are not so well normalized. On the other hand, the requests to update data (UPDATE, INSERT statements) are executed much faster on a relational databases, since a given information needs to be updated at only one place, unlike the NoSQL-based ones. In general, it can be concluded that with the traditional databases, normalization is highly recommended, while with NoSQL databases, the lack of normalization can be a sign that a good data model has been created. Researchers who came to similar conclusions are [2].

When creating a data model with the traditional databases, the approach is to take the initial data, then the work that is done is in the direction to define and decompose granular groups of objects that have common characteristics, and subsequently a separate table with the characteristics of the object is created. Relations between individual objects are defined by relations (1-1, 1-many, many-many). After that, when the application runs, the individual objects (tables) are combined into larger tables, and each of them contains information about several objects at once achieved through joins (JOIN operators). These combined tables are called lookups, or retrieval queries. They are the product of objects decomposed into separate tables, which are combined in order to extract common information about the relationship between several objects. An example of this is a database for books. One object is obviously a Book (with attributes Title, Price, ISBN, and Number of Pages). The other object could be Author (with characteristics First Name, Middle Name, Last Name, and Date of Birth). When implementing this model through the relational model, it represents a many-to-many relationship. This type of relationship also requires an additional table BooksFromAuthor. After that, by means of 2 JOIN operators, a reference such as Titles of books written by a certain author can be retrieved, which is a product that represents a table, and it is obtained by combining data from the tables Book and Author and BooksFromAuthor. While the organization of the data in this way saves space (there are no duplications in the Authors nor in the Books tables), data extraction is done by the computationally expensive join operator JOIN, applied twice in combination with the GROUP_CONCAT function, or its analogue (if the DBMS is different from MySQL), making it twice as slow.

From the point of view of the implementation of such a data model by means of NoSQL DBMS, the opposite approach is applied - First, the joins that are needed are defined, in this case a reference such as Titles of books written by an author, their price and ISBN number. A table is then created in the database for each such a join.

The arrangement made in this way can be realized by making a table BooksByAuthor, in which the first column is the name of the

book, and the second - a list of the names of the authors. In this scenario, in order to change the name of one of the authors, the entire record that contains the list of the names of all authors must be changed. This can lead to creating a source code of the information system that is difficult to write and maintain in the future. Fortunately, Cassandra CQL provides an easy way to modify a list of values. For example, a user can update the record for the book "The Last Thing He Told Me: A Novel" and add author "Laura Dave" in case the author is omitted as follows:

```
UPDATE BooksByAuthor SET Authors = Authors + {,Laura Dave'}
WHERE ISBN = ,978-1501171345';
```

Here the ISBN field is used as a natural key, but the table index could be used instead.

Separately, the Author and Book tables will exist so that all author names and all book names can be looked up. Obviously, when changing the author's name, it must be changed in the Author table as well as in the BooksByAuthor table that is, in 2 places, which leads to slower execution of UPDATE, INSERT and DELETE requests. In some information systems, some of the characteristics of objects in the data change very rarely or never (the Author of a book does not change because it is already printed), but at a minimum such information systems must support the ability to edit the data (for example incorrect initial entry and subsequent need for editing). NoSQL DBMS rely on exactly this, as well as on the optimization, if necessary, to retrieve the data with a JOIN request from the table, so that they are executed faster.

On the other hand, a NoSQL data model can also be implemented entirely through a relational DBMS, adding loosely normalized tables to the model to serve as the basis for frequently executed retrieval queries. The opposite - implementing a relational model using Cassandra is not possible because Cassandra does not support the JOIN operator and is not optimized to support the relational model.

3. Air quality reporting system

The GAIA air quality monitoring stations use high-tech laser particle detection sensors to measure the PM2.5 pollution, one of the most harmful air pollutants in real-time. They are very easy to set up and only require a WIFI access point and a USB power supply. Once connected, the air pollution levels are monitored and reported instantly in real-time on an Internet-based mapping application.

Regarding the system, the relational data model could look like this:

```
CREATE TABLE SENSOR STATION_INDEX INTEGER, SENSOR_DESCRIPTION TEXT (35), SENSOR_LONGTITUDE FLOAT, SENSOR_LATITUDE FLOAT, LAST_UPDATE_TIME TIMESTAMP;
CREATE TABLE READINGS STATION_INDEX INT, PARAMETER TEXT (35), VALUE INT, DATATIME TIMESTAMP;
CREATE TABLE HISTORY STATION_INDEX INT, PARAMETER TEXT (35), VALUE INT, DAY DATE;
CREATE TABLE FORECAST STATION_INDEX INT, PARAMETER TEXT (35), VALUE INT, DAY DATE;
```

The PARAMETER field can have one of the following values: PM25, O3, NO2, CO2, SO2, TEMERATURE, RH, PRESSURE, HUMIDITY, WIND_SPEED, WIND_DIRECTION.

This database model is fed with standard real CSV (Comma Separated Values) format from the https://www.londonair.org.uk web service. They can easily be imported into the data model constructed this way, and are compatible with it.

In [3] is investigated the applicability of the NoSQL databases in the research of Air Quality Index, by using MongoDB. On the other hand, in the document [4] several steps that lead to the construction of a NoSQL data model are outlined. These are: Step 1: Build the application workflow; Step 2: Model the requests required by the application; Step 3: Create the tables; Step 4: Create the primary key; Step 5: Effective use of basic data types.

Step 1. The main screens of the application must be identified, as well as the data that is required for each screen.

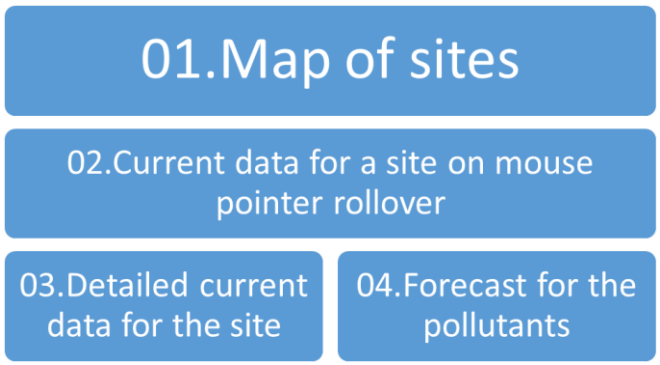


Figure 1. Data, required by each application screen

The sequence of the workflow steps is important because it helps to determine what data is available and required for each screen. For example, before the detailed information about a station can be displayed (User Step 03 above), a station identifier (StationIndex) is required. Likewise, to display a forecast for a station (User Step 04), its ID is required.

Step 2. A diagram of the interrelationships in the application is prepared, and at this step the attention could be returned back to the previous Figure 1 to correct it.

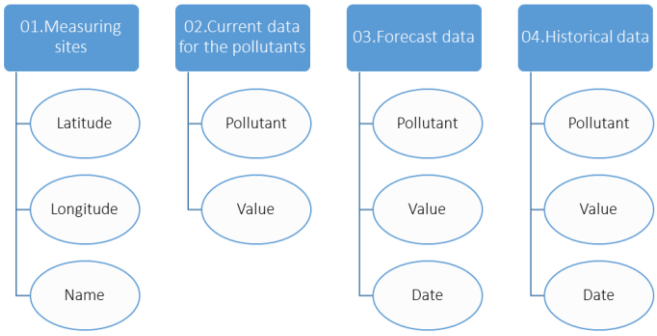


Figure 2: User steps and modeling of the requests when working with the application.

Step 3. Create the tables. The next step is to structure the tables to support the required queries of the application.

Tables are designed to match the individual views /screens/ of the application. In general, the application has the following screens:

Initially, a world map is loaded, with POI (Point of Interest) markers marking the various measuring stations around the world. To be fed with data, this screen needs only one table that contains the coordinates of all the measurement points, as well as a key to the table with the data of all measurements for a given point.

If the table is called StationCoordinates, then it will have the columns StationIndex, Latitude, Longitude, StationName.

Another screen with data appears when the mouse cursor is positioned over a measuring station. Then a request is sent to the database to load the data for the measuring station in question. The table that supports this screen should contain the data for the current values of the indicators and should have columns: index of the measuring station - StationIndex, AQI name, forecast for the next 3 days of the AQI indicator. Historical data for indicators PM25, PM10, NO2, SO2, CO, for the last 12 months, can be stored in the LastYear table with columns StationIndex, Indicator, Date, Value,

and retrieved with a separate query to the database when the screen loads.

It is beyond doubt that all air quality sensors (AQS) around the world generate a huge amount of relatively well-structured information, over large periods of time. Therefore, when working with a lot of data, the advantages of one or the other technology will become even more obvious. From this point of view, it is of interest to build a relational and NoSQL model of the data and to compare their performance in different scenarios - writing, editing and extracting data.

4. Forecast

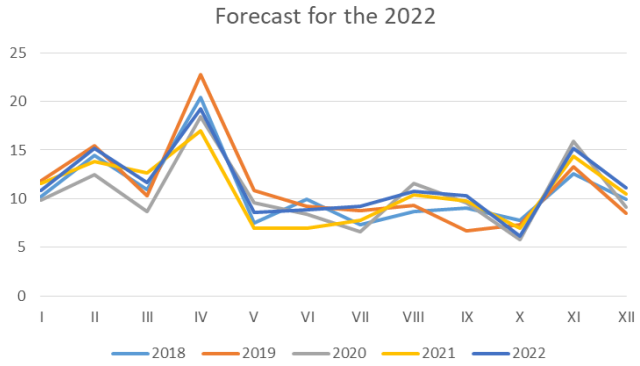
Contamination prediction is of particular importance to the users of such a system. It is needed by the citizens who are interested, as well as by the state administration, which could introduce social incentives, for example, higher parking fees in the polluted areas, or a cheaper or even free ticket for the public transport in a case of a bad trend. The accumulated information can help in making such a forecast. The aim is to test how well it is possible to predict days with poor air quality indicators. On colder days, the burning of solid fuel sources can be expected to increase, and combined with a lack of wind, can lead to smog in a given area. Unfortunately, the cleanliness of the air also depends to a large extent on the weather forecast. The region that is studied also has an impact, as forecasts may differ for different regions. The area for which the research is being done is the center of the city of London in Great Britain, because the indicators of air pollution have been measured and recorded there for a long time, and also the information is freely available. Other researchers are also examining data about this city [5], [6]. For such a forecast, also the statistical methods can be used (linear regression, following the example of scientists [7]), as well as neural networks (time series of indicators, as the following researchers have done [8], [9] and [10]). Statistical data for past periods as input parameters of the model could be accepted – what air pollution was in the past when the conditions were similar, and their development over time.

The aim is to extract some general trend in the data by which to impose restrictions on the population, which could be ban on cars, high bills for heating during the dirty period, a ban on burning coal, high electricity bills, an increase in the excise duty on fuels, stoppage of warm water, etc. as this could lead to an improvement in the air quality during the period in question.

Table 1 contains monthly average data for PM 2.5 in (ug/m3) taken from [11] for the period 2018-2021. In the last column, the values of this indicator for the year 2022 are predicted by the means of a neural network with a radial basis function. The error obtained during the preparation of the forecast is within 1.51%. The error is calculated in percentage as a deviation when comparing the forecast for 2022 with the actual data from it.

Month	Year				
	2018	2019	2020	2021	2022-forecast
I	10.23	11.81	9.89	11.52	10.83
II	14.42	15.41	12.47	13.83	15.21
III	10.95	10.32	8.68	12.62	11.64
IV	20.38	22.74	18.41	17.02	19.25
V	7.54	10.84	9.55	6.93	8.55
VI	9.98	9.26	8.40	6.99	8.85
VII	7.31	8.76	6.64	7.78	9.23
VIII	8.65	9.27	11.52	10.43	10.73
IX	9.02	6.71	9.54	9.73	10.31
X	7.81	7.35	5.74	6.93	6.18
XI	12.58	13.28	15.9	14.33	15.21
XII	9.95	8.46	9.15	10.45	11.13

Table 1. Average data for the period 2018-2021 and the forecast for the 2022 for the air pollution indicator PM 2.5



Graph 1. Forecast for PM2.5 for the 2022

The graph shows that, in general, the air pollution with PM 2.5 particles reaches its highest value in the month of April, from May to October it moves in more moderate limits, and then during the cold winter months it starts to increase until the month of April.

The authors [12] also investigated the prediction of air pollution by regression analysis. The bivariate analysis (correlation matrix) with the Pearson's coefficient allows to evaluate both the prediction obtained and the interrelationships between the data [13]. In general, it can be concluded that every year the trend of PM 2.5 particle pollution is uniform, which is repeated every year. That is, to a certain extent, we can talk about the predictability of the pollution in question.

Year	2018	2019	2020	2021	2022
2018	1.000	0.934	0.852	0.881	0.931
2019	0.934	1.000	0.853	0.806	0.885
2020	0.852	0.853	1.000	0.860	0.926
2021	0.881	0.806	0.860	1.000	0.954
2022	0.931	0.885	0.926	0.954	1.000

Table 2. Correlation matrix with the Pearson coefficient.

Table 2 shows that there is a high degree of correlation both between the individual variables used for forecasting /the data for the years 2018-2021/, and between them and the forecast for 2022 prepared by the neural network.

5. Conclusion

The authors [14] investigate the performance of different DBMS when running on devices with reduced power consumption and large amounts of data. For these devices, the speed is critical because the devices in question have limited resources. To evaluate the performance of the different data management systems, the neural network prediction process was run twice, in one case using the Python Datastax driver to connect to the Cassandra database, and in the second case using the MySQL Connector driver to connect to MySQL. In both cases, the time to prepare the forecast was recorded, and in the first case it was prepared in 4704 milliseconds, and in the second in 3987 milliseconds. The data was extracted raw from the database and averaged at runtime to get a better idea of which of the two systems would be faster at the extraction itself. The data covers a 4-year interval and is only for the PM2.5 metric measured every 15 minutes, or about 143,080 records that include date, time, and metric value. Keras library was used for preparing the neural network forecast.

6. References

1. Besmir Sejdiu, Florije Ismaili, Lule Ahmedi, Computers 2021, 10(10), 127; https://doi.org/10.3390/computers10100127, IoTSAS: An Integrated System for Real-Time Semantic Annotation and Interpretation of IoT Sensor Stream Data

2. Jeang-Kuo Chen, Wei-Zhe Lee, Algorithms 2019, 12(5), 106; https://doi.org/10.3390/a12050106, An Introduction of NoSQL Databases Based on Their Categories and Application Industries

3. The Impact of Bike-Sharing Ridership on Air Quality: A Scalable Data Science Framework,

4. Nina Hua, Victoria Suarez, Rebecca Reilly, Philip Trinh, Paul Intrevado, Diane Myung-kyung Woodbridge, Data Science Program, University of San Francisco, 2019, IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, Data Modeling in Apache Cassandra, Five Steps to an Awesome Data Model

5. Roba Zayed, Maysam Abbod, DataSCI, vol. 4, no. 2, pp. 5-10, Jan. 2022, Big Data AI System for Air Quality Prediction

6. Nadia N. Sánchez-Pozo, Sergi Trilles-Oliver, Albert Solé-Ribalta, Leandro L. Lorente-Leyva, Dagoberto Mayorca-Torres, Diego H. Peluffo-Ordóñez International Conference on Hybrid Artificial Intelligence Systems, HAIS 2021: Hybrid Artificial Intelligent Systems pp 293–304, Algorithms Air Quality Estimation: A Comparative Study of Stochastic and Heuristic Predictive Models

7. Akram Jamal, Ramin Nabizadeh Nodehi, Journal of Air Pollution and Health (Quarterly), eISSN: 2476-3071, Tehran University of Medical Sciences, Tehran, Iran, 2017, Predicting Air Quality Index Based on Meteorological Data: A Comparison of Regression Analysis, Artificial Neural Networks and Decision Tree

8. Heidar Maleki, Armin Sorooshian, Gholamreza Goudarzi, Zeynab Baboli, Yaser Tahmasebi Birgani & Mojtaba Rahmati, Clean Technologies and Environmental Policy volume 21, pages 1341–1352 (2019), Air pollution prediction by using an artificial neural network model

9. Sean Mc Grath, Eoin Garrigan, Liaoyuan Zeng, 2021 IEEE International Conference on Electronic Technology, Communication and Information (ICETCI), Predicting Air Quality Index Using Deep Neural Networks

10. Fabio Cassano, Antonio Casale, Paola Regina, Luana Spadafina & Petar Sekulic, International Symposium on Ambient Intelligence, ISAmI 2019: Ambient Intelligence – Software and Applications, 10th International Symposium on Ambient Intelligence pp 36–44, A Recurrent Neural Network Approach to Improve the Air Quality Index Prediction

11. https://www.londonair.org.uk/

12. José Ángel Martín-Baosa, Luis Rodriguez-Benitez, Ricardo García-Ródenas, JunLiu, Applied Soft Computing, Volume 115, January 2022, 108282, https://doi.org/10.1016/j.asoc.2021.108282, IoT based monitoring of air quality and traffic using regression analysis

13. Ning Jin, Yongkang Zeng, Ke Yan, Zhiwei Ji, IEEE Transactions on Industrial Informatics (Volume: 17, Issue: 12, December 2021), Page(s): 8514 – 8522, Multivariate Air Quality Forecasting With Nested Long Short Term Memory Neural Network

14. Piotr Grzesik, Dariusz Mrozek, International Conference on Computational Science 2020: Computational Science – ICCS 2020 pp 371–383, Comparative Analysis of Time Series Databases in the Context of Edge Computing for Low Power Sensor Networks