23rd International Conference on Renewable Energies and Power Quality (ICREPQ'25)
Tenerife (Spain), 25th to 27th June 2025
ISBN-13: 978-84-09-66511-2

# No Structured Query Language database for digital twin support of wind turbine farms

F. Herrera[1], S. Esteban[2] and M. Santos[3]

[1] Dept. Arquitectura de Computadores y Automática
Computer Science Faculty
Universidad Complutense de Madrid
28040-Madrid, España

[2] Dept. Arquitectura de Computadores y Automática
Faculty of Physics
Universidad Complutense de Madrid
28040-Madrid, España

[3] Institute of Knowledge Technology
Computer Science Faculty
Universidad Complutense de Madrid
28040-Madrid, España

**Abstract.** Offshore wind turbines are emerging as a key technology to provide energy through a still not fully exploited natural resource such as the ocean winds. However, offshore wind turbines present more challenges than their onshore counterpart due to their location in a more unstable environment. A technology that helps to improve the control and monitoring of these turbines is a digital twin. The aim of this work is to design a database able to represent and manage the digital twins of wind turbines farms. This paper explains a No Structured Query Language, from now on referred as NoSQL, approach that can handle efficiently all the data needed to have an accurate digital representation of a wind turbine farm, storing documents for the wind farms, the turbines, the signals sent by the turbines, and the control and error detection functions. This is essential to setting a good base to develop a full digital twin control on their physical counterparts.

**Key words.** Offshore wind turbine, Digital twin, Control, Monitoring, No Structured Query Language NoSQL.

## 1. Introduction

The world is currently in the midst of an energy crisis caused by two key factors, the environmental impact of energy production and the availability of energy. The various renewable energies are globally accepted as the answer to the first of these issues [1]. However, they still do not have the capacity to completely replace traditional energy sources, which forces countries that do not have extractable natural energy resources, such as Spain, to import energy from other nations. This generates situations of dependence that are not harmful in times of prosperity or stability but can become a manifest weakness when this situation is modified, as has been demonstrated in recent years within Europe with the purchase of Russian gas.

Within this framework, one of the main sources of renewable energy is wind, and especially offshore wind, which is expected to grow the most in the coming years [2]. In order to reach a point where it can be supplied exclusively by renewable energy sources, it is necessary to make optimal use of the generating infrastructures.

This is where developing digital twins can play a decisive role [3] [4]. A digital twin is the virtual representation of a physical object, which is fed in real time with data from its physical counterpart [5]. In addition, compared to a simple telemetry record, a digital twin contains both historical wind turbine parameters and historical telemetry data. This allows for real-time evaluation of wind turbine performance, allowing anomalies to be detected by comparing telemetry data with simulations of the parameterized twin. Furthermore, model parameters could also be estimated in real time; the evolution of these parameters would be useful for predictive maintenance.

A good monitoring and control of the turbines allows an increase in the production of each wind turbine, as well as their useful life and a drastic reduction in their maintenance costs [6]. These improvements are accompanied by the ability to deploy turbines in areas previously impossible due to adverse weather conditions [7]. The inaccessibility of these areas aggravate the need for further monitoring and predictive maintenance.

A good architecture for digital twinning of wind turbine farms should be designed to enable real-time monitoring,

predictive maintenance, performance optimization, and seamless integration with other systems. It should allow:

- Real-time data collection: This includes information on the turbine stats, such as rotational speed, power generated, blade angle.... In addition, it will need to collect environmental data from geographic coordinates, such as local weather conditions [8]. These data will be used to keep the digital twin of the farms updated [9].
- Data storage: All data collected should be easily accessible, this will allow analysis of historical data, which is valuable in order to improve turbine and farm performance.
- Farm management: Data from the wind turbines of a given farm should be easily accessible which will facilitate the joint management and monitoring of these turbines.
- Flexibility in storage: An essential feature is the ability to store various control strategies, error detection functions and heterogeneous turbine models.
- Scalability for large wind farms. A scalable digital twin architecture for wind farms must handle massive data volumes from hundreds of turbines while maintaining real-time performance.

The structure of the paper is as follows. In the next section 2, we will give the reasons for choosing a database system over others and show the structure of the database. Section 3 describes in depth the components of the database and its objectives. Section 4 presents a test case to validate the proposal. The paper ends with the conclusions and future works.

## 2. Choosing the Database System

Once the requirements for a good database have been defined, the most appropriate model must be found. The solution proposed in this work is a NoSQL document-oriented database. These databases are specialized in managing and accessing large volumes of data, as well as providing a more flexible structure that allows storing different types of data in the same place, characteristics that are not found in traditional databases.

Specifically we are going to use MongoDB which is a database system that has support in the main distributions of Linux, Windows and macOS.

In MongoDB the information is organized as follows. First, there is the MongoDB server, which can host multiple databases. Each of these databases contains collections, which are analogous to tables in a relational database and are created with a specific name within the database. These collections store documents that generally share common fields. These fields take the form of key-value pairs and can be of various types, such as integers, arrays, dates, or other embedded documents.

Other essential features of MongoDB are:

- The possibility of selecting indexes in the collections to create auxiliary structures based on balanced search trees and achieve efficiency in reads, with almost no cost in writes.
- Sharding that distributes data into separate sets based on selected indexes for load balancing. Allowing a horizontal scaling of the server to be performed. This possibility makes a MongoDB-based system that is already faster in itself than a SQL one, to be even faster for large volumes of data [10],
- Finally, one last feature to highlight is the time series collections. This is a type of collection that serves to efficiently store data that is obtained recurrently over time.

These features make it a strong choice for digital twin architectures. A database for MongoDB is proposed that follows a hybrid structure, merging elements from relational and non-relational databases to take advantage of their specific characteristics. This database consists of six collections: one for wind turbines, one for farms, one for signals, and three additional collections for controls: one applicable to turbines, one to farms, and one for fault detection systems.

The diagram shown in Fig. 1 adopts the crow's foot notation used in relational databases, since the proposed solution will use document attributes as foreign keys as in relational databases, and will ensure compliance with this structure, leaving the flexibility of NoSQL in those document fields designated as other documents.

The diagram shows the main modules of the DT. First, the wind turbine and turbine farm, and the respective control systems (WT control and farm control). The signal block supplies power to all modules and collects information. Some of these signals are also used for fault detection.

## 3. Database Collections

### *A. Control and fault detection functions*

These collections store information about the controls for the turbines, for the farms and the information for the fault detection functions. Each document represents the template with the parameters of a function with all the fields it needs and their default values. These collections benefit greatly from NoSQL, which allows to store documents that are going to have very different fields, which could not be done in a classical relational database.

The attributes that represent a document in these collections are:

1) *id:* It is a string that stores the name of the function.
2) *parameters:* This optional field is an object that contains all the fields necessary for the correct operation of the particular function.

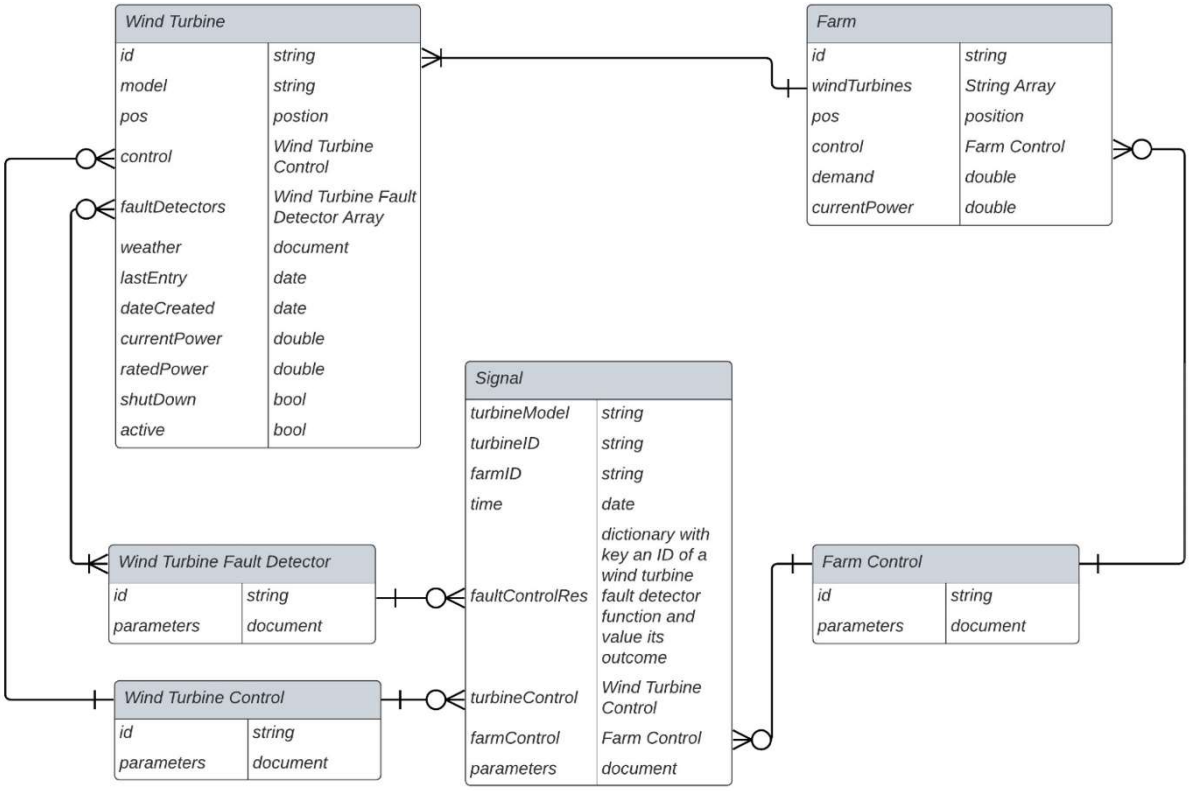


*Fig. 1 Database diagram*

Fig. 2 shows different examples of documents that would go into the collection of turbine controls module. Specifically, it can be seen how "parameters" can vary greatly depending on the controller that is wanted to be applied.

PID Control Document

```
{
  "id": "PIDControl",
  "parameters":{
    "pitchPID"{
      "kp": 0.07,
      "kd": 0.03,
      "ki": 0.01
    },
    "loadPID"{
      "kp": 0.003,
      "kd": 0.01,
      "ki": 0.01
    },
    "referenceRPM": 500,
    "maxLoad": 15,
    "minLoad": 0,
    "maxPitch": 30,
    "minPitch": 15
  }
}
```

Manual Control Document

```
{
  "id": "manualControl",
  "parameters":{
      "pitch": 0.07,
      "load": 0.03,
  }
}
```

Embedded Control Document

```
{
  "id": "embeddedControl",
  "parameters": null
}
```

*Figure 2 Examples of Wind Turbine Controls Documents*

## B. Signals

In this collection the signals received by the turbines are stored. Given the nature of this data, it will be a *time series collection*. This collection also makes use of NoSQL properties that allow storage of heterogeneous data. Wind turbine designs vary and, therefore, it is also to be expected that they will have different sensors and that, although there is data common to all turbines such as rotor revolutions, there is other data that will be unique to certain models.

The choice was made to create a collection of signals instead of including the signals from each turbine in the document associated with the turbine. This choice is based on the fact that keeping the signals in their own collection favors the joint analysis of all signals. Let us imagine that we want to analyze all the signals recorded at a specific time due to certain atmospheric conditions, or that we want to perform an exhaustive analysis of a particular type of control. It will be much more efficient to query a unified collection than to have to go through each turbine document looking for the signals that meet the conditions.

Furthermore, in case of embedding the signals in the turbine documents there would be situations where a turbine document would end up being too heavy, which could give performance problems, because the processing of a single document cannot be scaled horizontally. Whereas by relegating the signals to their own collection, scaling can be performed without problems.

The attributes that each signal document has are:

1) *time:* The date according to ISO 8601 standard.
2) *model:* String that identifies the turbine model. This field also acts as an index.
3) *turbineID:* This is the string identifier of the turbine that sent the signal. This field also acts as an index.
4) *farmID:* This is the identifier in string format of the farm that sent the signal. This field also acts as an index.
5) *turbineControl:* This field is a document of the turbine control collection type that indicates the type of control that was active on the turbine when the signal measurement was taken. This field also acts as an index.
6) *farmControl:* This field is a document of the farm control collection type that indicates the type of control that was active on the farm when the signal measurement was taken. This field also acts as an inde.
7) *faultControlRes:* This field stores a dictionary with key the identifier of a fault detection function and value the parameters that the function had for that turbine, as well as the value that the function calculated.
8) *parameters:* This field is an embedded document containing the various fields of the signal. An example for an offshore turbine would contain the pitch, the load, the rpm, the accelerations on the shafts....

### C. Wind turbines

Each turbine document contains the following attributes:

1) *id:* This is a string that stores the turbine identifier. This field also acts as an index.
2) *model:* This is a string that identifies the turbine model. This field also acts as an index.
3) *pos:* This attribute is the position in coordinates of the turbine, and is represented by a pair of integers: latitude, longitude.
4) *control:* The field contains the information related to the control applied by the turbine. It follows the same format as the documents in the turbine control collection.
5) *faultDetectors*: rAray of Wind Turbine Fault Detector type documents. It contains all the fault detection functions that will be used to analyze the turbine signals when received at the digital twin.
6) *Weather:* The weather conditions at the turbine location. It is an embedded document that shows the weather conditions found at its geographic location.
7) *lastEntry:* A timestamp showing when the last signal was received from the turbine. This JSON object contains the year, month, day, hour, minute and second when the signal was received.
8) *dateCreated:* The date the turbine was created. Same format as "lastEntry".
9) *currentPower:* The amount of power generated by the turbine at that moment is a number in float format.
10) *ratedPower:* The rated power of the turbine is a number in float format.
11) *shutDown:* A Boolean that indicates whether the turbine is available for operation.
12) *active:* A Boolean that indicates whether the turbine is currently running or not.

### D. Wind turbine farms

This collection stores information about all wind turbine farms. Each farm document contains the following attributes:

1) *id:* This attribute is a string indicating the name of the farm. This field also acts as an index.
2) *pos:* This field follows the same structure as the geographical position of the turbines.
3) *windTurbines:* This attribute is an array with the identifiers of the turbines that are part of the rook.
4) *control:* The field contains the information related to the control applied by the farm. It follows the same format as the documents in the farm control collection.
5) *currentPower:* It is the energy produced by the farm at that moment, it is a number in float format.
6) *Demand:* The demand received by the farm is a number in float format.

The turbine documents could have been embedded in this collection, but we have chosen to store only the identifiers of the turbine documents as would be done in relational databases. This was done for the same reason that the signals were not embedded, to avoid performance problems when the farm document has to be updated whenever one of its turbines receives an update.

## 4. Test Case

With a reduced design as shown above, the test architecture shown in *Fig.* 3 has been developed. It can be seen that, on the one hand, wind turbine prototypes communicate with an API in charge of managing the database. On the other hand, there is a graphical interface that allows interacting with the API for monitoring and controlling the wind turbines. Both the API that manages the requests, and the graphical interface for the digital twin have been developed using JavaScript.

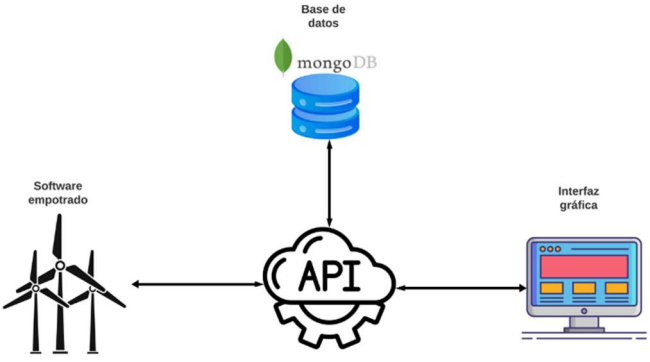


*Figure 3 Schematic diagram of the test structure*

A key aspect of this database approach is the ability to apply new controls without requiring on-site modifications. By structuring the system in a way that allows the creation of new control functions, it becomes easier to integrate emerging technologies and optimize performance without disrupting operations. This flexibility enhances the digital twin performance compared to traditional database models by enabling rapid adaptation to new control strategies.

To validate the architecture, we tested it by feeding the database with data from small-scale wind turbine prototypes in the laboratory, as well as from simulated wind turbines using a specific high-fidelity simulation software of wind turbines called OpenFast, by NREL (National Renewable Energy Laboratory, USA). *Fig. 4* illustrates the start-up process of a prototype, as visualized in the graphical interface using data from its digital twin. The wind turbine was controlled using a PID control system executed on the server, with the parameters displayed in *Fig. 2*


*Figure 4 Start-up of a wind turbine prototype*

*Fig. 5* presents a sample document generated for one of our first prototypes.


```
_id: ObjectId('64fb4162aae63eda2d015ea7')
▸ pos: Object
  weather: null
▾ control: Object
    id: "InternalAutoControl"
  ratedPower: 500
  currentPower: 0
  active: false
▾ dateCreated: Object
    año: 2023
    mes: 11
    dia: 20
    hora: 17
    minuto: 42
    segundo: 10
▾ lastEntry: Object
    año: 2023
    mes: 11
    dia: 21
    hora: 16
    minuto: 42
    segundo: 42
  shutDown: false
  model: "Prototype1"
  id: "199954115615177"
```


*Figure 2 Wind turbine document*

## 5. Conclusions and Future Work

This article presents the reasons why choosing a document database for the representation of wind turbine farms is a key point to improve their operation and maintenance.

In addition, the advantages of keeping in the same collection the signals sent by the turbines have been described, orienting the design to the analysis of historical signals. The capacity of NoSQL and the design to include collections that store the parameters for heterogeneous control and error monitoring functions have also been highlighted.

The model could also be adapted for PV panel installations by modifying the collections to accommodate different types of energy generators rather than just wind turbines. This can be achieved by introducing a parameter that specifies the type of energy generator. Additionally, the farm collections would be adjusted to include separate control documents for wind turbines and PV panels, ensuring tailored management for each energy source.

Additionally, specific parameters for monitoring energy production, panel efficiency, and fault detection can be incorporated. The system can also be enhanced to include predictive maintenance based on environmental conditions and degradation patterns.

The next proposed steps would be to extend and improve the development of the test framework to cover all the issues raised in the database design and hardware implementation [10].

## Acknowledgement


This work has been partially funded by the Ministry of Science and Innovation of Spain, under the project MCI/AEI/FEDER number PID2021-123543OBC21.


## References


[1] REN21. “Renewables 2023. Global Status Report. A comprehensive annual overview of the state of renewable energy. https://www.ren21.net/gsr-2023/ (2023).

[2] M. Tomás-Rodríguez, and M. Santos. "Modelado y control de turbinas eólicas marinas flotantes." Revista iberoamericana de automática e informática industrial 16, no. 4 (2019): 381-390. https://doi.org/10.4995/riai.2019.11648

[3] M. Wang, C. Wang, A. Hnydiuk-Stefan, S. Feng, I. Atilla, and Z. Li. "Recent progress on reliability analysis of offshore wind turbine support structures considering digital twin solutions." Ocean Engineering 232 (2021): 109168. https://doi.org/10.1016/j.oceaneng.2021.109168

[4] F. Rodríguez, W.D. Chicaiza, A. Sánchez, and J. M. Escaño. "Updating digital twins: Methodology for data accuracy quality control using machine learning techniques." Computers in Industry 151 (2023): 103958. https://doi.org/10.1016/j.compind.2023.103958

[5] F.J. Pimenta, F., J. Pacheco, C. M. Branco, C. M. Teixeira, and F. Magalhães. "Development of a digital twin of an onshore wind turbine using monitoring data." In Journal of

Physics: Conference Series, vol. 1618, no. 2, p. 022065. IOP Publishing, 2020. DOI 10.1088/1742-6596/1618/2/022065

[6] O. O. Olatunji, P. A. Adedeji, N. Madushele, and T.-C. Jen. "Overview of digital twin technology in wind turbine fault diagnosis and condition monitoring." In 2021 IEEE 12th International Conference on Mechanical and Intelligent Manufacturing Technologies (ICMIMT), pp. 201-207. IEEE, 2021. DOI: 10.1109/ICMIMT52186.2021.9476186

[7] Z. Bowen, Z. Zhang, G. Li, D. Yang, and M. Santos. "Review of Key Technologies for Offshore Floating Wind Power Generation." Energies 16, no. 2 (2023): 710. https://doi.org/10.3390/en16020710

[8] R. Pandit, D. Infield, and M. Santos. "Accounting for environmental conditions in data-driven wind turbine power models." IEEE Transactions on Sustainable Energy 14, no. 1 (2022): 168-177. DOI: 10.1109/TSTE.2022.3204453

[9] R. Pandit, D. Astolfi, J. Hong, D. Infield, and M. Santos. "SCADA data for wind turbine data-driven condition/performance monitoring: A review on state-of-art, challenges and future trends." Wind Engineering 47, no. 2 (2023): 422-441. https://doi.org/10.1177/0309524X2211240

[10] Y. -S. Kang, I. -H. Park, J. Rhee and Y. -H. Lee, "MongoDB-Based Repository Design for IoT-Generated RFID/Sensor Big Data," in IEEE Sensors Journal, vol. 16, no. 2, pp. 485-497, Jan.15, 2016, doi: 10.1109/JSEN.2015.2483499.

[11] I. Tajadura, J.E. Sierra-García, and M. Santos. "Communication library to implement digital twins based on matlab and IEC61131." In APCA International Conference on Automatic Control and Soft Computing, pp. 262-271. Cham: Springer International Publishing, 2022. https://doi.org/10.1007/978-3-031-10047-5_23