Article

SQL and NoSQL Databases in the Context of Industry 4.0

Vitor F. de Oliveira*, Marcosiris A. O. Pessoa, Fabrício Junqueira and Paulo E. Miyagi

Universidade de São Paulo , Escola Politécnica, São Paulo, Av. Professor Mello Moraes, 2231 – Butantã, 05508-030 São Paulo - SP, Brazil.; marcosiris@usp.br (M.A.O.P.); fabri@usp.br (F.J.); pemiyagi@usp.br (P.E.M.)

* Correspondence: vitor.furlan@usp.br; Tel.: +55 (11) 96033-7582

Abstract: The data-oriented paradigm has proven to be fundamental for the technological transformation process that characterizes Industry 4.0 (I4.0) so that Big Data & Analytics is considered a technological pillar of this process. The literature reports a series of system architecture proposals that seek to implement the so-called Smart Factory, which is primarily data-driven. Many of these proposals treat data storage solutions as mere entities that support the architecture's functionalities. However, choosing which logical data model to use can significantly affect the performance of the architecture. This work identifies the advantages and disadvantages of relational (SQL) and non-relational (NoSQL) data models for I4.0, taking into account the nature of the data in this process. The characterization of data in the context of I4.0 is based on the five dimensions of Big Data and a standardized format for representing information of assets in the virtual world, the Asset Administration Shell. This work allows identifying appropriate transactional properties and logical data models according to the volume, variety, velocity, veracity, and value of the data. In this way, it is possible to describe the suitability of SQL and NoSQL databases for different scenarios within I4.0.

Keywords: Industry 4.0; Database; Data models; Big Data & Analytics; Asset Administration Shell

1. Introduction

Industry 4.0 (I4.0) designates the technological transformation process in production systems, logistics, and business models observedsince the last decade [1]. The integration of digital technologies has promoted changes in the development phase [2,3]; flexibility of production [3,4]; efficiency in the use of resources [5,6]; level of automation and digitalization of the organizations [7,8]. Data emerged as a fundamental resource for these transformations due to its characteristics such as low cost, apparent inexhaustibility, and the possibility of cost reduction and value creation [9]. Thus, it is clear the importance of the data-oriented paradigm in the context of I4.0 [9].

System architectures are used to promote the integration of digital technologies that enable the use of data for industrial innovations [10–12]. While there is concern about optimizing these architectures in many respects, the impact of databases on their performance is not always taken into account. It is possible to observe that, in many cases, databases are treated as mere entities that support the functions of architectures. Therefore, this paper proposes the identification of data models that better suit different scenarios in the context of I4.0. This phenomenon is characterized by its key enabling data-related technologies and methods so that a consistent description of the nature of the data in this context could be achieved. By identifying the advantages and limitations of SQL and NoSQL data models for such data characteristics, it is possible to discuss the suitability of these models for different scenarios in the context of I4.0.

2. Materials and Methods

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

Presenting the context in which this paper is inserted is essential to justify the choice of materials and methods adopted in the work from which it derives. For this reason, this section begins with a contextualization of Industry 4.0. Understanding some of its main characteristics is important to justify the relevance of this paper to (i) demonstrate the gap that exists in system architectures for I4.0 concerning data storage solutions used by these systems; and (ii) identify patterns, methods, and technologies whose relevance to I4.0 is such that, from them, it is possible to obtain a characterization of the data in this context. Thus, this information is combined to propose database suggestions for the I4.0 context.

There is a consensual understanding nowadays that manufacturing automation systems have been undergoing a continuous transformation of technological paradigms since the last decade [1]. Authors claim that these transformations, obtained from integrating a series of independent digital technologies and a certain degree of independence from each other, configure Fourth Industrial Revolution [13]. Because of the global scale of these changes, several initiatives worldwide, such as the *Plattform Industrie 4.0*[1], the Industrial Internet Consortium[2], and the Standardization Council Industrie 4.0[3], seek to establish guidelines for this process of technological transformation. The need to have a guide (or multiple equivalent guides) for the technological transformation process associated with the Fourth Industrial Revolutin is due to the fact that, unlike the first three, the Fourth Industrial Revolution was identified as such already in its early stages. Thus, these initiatives become responsible for outlining the advancement of technological transformation in manufacturing, proposing a common understanding of the phenomenon, establishing standards, and so on.

Among the different technological aspects mentioned above, some are highlighted in this work, and focus is given to the so-called I4.0, a term often used as a synonym for the Fourth Industrial Revolution. In Germany, the *Plattform Industrie 4.0* was created, a consortium of various organizations, including industries, universities, and the German government, proposing to shape the digital transformation in manufacturing according to the precepts of I4.0. The meaning of the term "Industry 4.0" is the object of analysis by several researchers [14–16]. Instead of presenting a definition, the option is to describe the phenomenon in terms of its characteristics: I4.0 is characterized by production processes that quickly adapt to market demands and with effective interconnection between all entities involved in these processes. The result of this phenomenon is the conception of the so-called Smart Factory, which aims at manufacturing based on intelligent services and processes [17].

The main materials used in this work were technical publications and academic works. Considering that I4.0 is the result of cooperation between academia, industry, and government organizations, it was impossible to use only literature review methods of academic publications.

Characterizing a system's data is essencial for choosing the database to be adopted in the architecture for this system. In this work, this characterization is made based on technologies, methods, and standards for data in I4.0. Other relevant features for database design are fundamentally application dependent and are beyond the scope of this paper which seeks to expand the coverage of its contributions. The following paragraphs describe the materials and methods adopted to characterize data in different scenarios in the context of I4.0 and identify which data models are more or less suitable for different scenarios.

Ensuring interoperability among systems is one of the requirements for implementing the Smart Factory [16,18,19]. For this purpose, the entities that establish guidelines for the advancement of I4.0 proposed a standard format for digitally representing and managing elements involved in carrying out productive activities – the Asset Administration Shell (AAS) – whose concept, structure, metamodel, and perspectives for implementation will be presented in subsection 3.2. Current works propose the imple-

[1] https://www.plattform-i40.de/PI40/Navigation/EN/Home/home.html
[2] https://www.iiconsortium.org/
[3] https://www.sci40.com/

mentation and use of the Asset Administration Shell in system architectures that seek to use data for different purposes. However, it is noted that less attention is paid to the design of the database to be used in these architectures. To confirm this statement, a systematic review of the literature was carried. The adopted procedures were the following:

- The paper databases used in the search were: Web of Science, IEEEXplore, Science Direct, Scopus, and Google Scholar;
- The following search string was defined to find papers: "Asset Administration Shell" AND "Database";
- It was observed that, among the selected databases, the only one to return a considerable number of papers was the Google Scholar, which included papers from the other databases and, therefore, was the only one used;
- The following keywords were defined for ranking the papers: "AAS"; "Asset Administration Shell"; "Database"; "DBMS" (database management system); "Implement"; "Implementing"; "Implementation"; "Storage".
- To rank the papers, it was decided that each keyword in the title of the article would assign 5 points to it (the Google Scholar platform does not allow exporting the abstract or keywords of the article);
- Papers with a score greater than or equal to 5 were classified as accepted, and their content was analyzed;
- It was researched which of the papers classified as accepted cited the implementation data model and/or DBMS used.

Considering that I4.0 is a process of technological transformation, important data-based digital technologies and methods were identified. Those have such importance for this process that a description of the nature of the data in the context of I4.0 can be obtained from them. In addition to academic works, technical publications such as working papers from key organizations and entities for I4.0 were also considered in this process.3. Basic Concepts

This section presents a theoretical framework composed of essential basic concepts for the work. Concepts that are related to database include SQL and NoSQL data models, transactional properties, and theorems regarding these properties. Also presented is the Asset Administration Shell concept, an artifact developed for the representation of Industry 4.0 components in the digital world.

3.1. SQL and NoSQL Databases

A logical and coherent collection of data with an intrinsic meaning forms a database [20]. A database store and ensure the persistence and integrity of data that represent assets, in addition to allowing these data to be made available to interested users. A database is created and maintained through a database management system (DBMS), a computer program that helps maintain and use data sets that compose the databases [21]. These programs have the following advantages: they enable efficient and concurrent access to data; ensure data integrity and security; protect against failures and unauthorized access; support multiple views of data; and finally, they guarantee independence, that is, the isolation between data and applications through data abstraction [21,22].

Data abstraction provides the independence between data and applications, occurs through data models. A data model is a set of concepts used to describe the structure of a database [20]. The logical data model describes data in such a level of abstraction that hides some details of the physical storage, which allows the end-user of the data to understand them. At the same time, as they are not so far from the low level, these concepts can be used directly to implement a database in a computer system. DBMSs are usually characterized by the logical data models they implement and, for this reason, this work focuses on this level of data abstraction.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

3.1.1. Relational / SQL data model

The relational data model was, for many years, the default choice for database implementation [23]. It uses the concept of "relation" in a mathematical sense to represent data. Instead of presenting a formal mathematical definition of the term, which can be found in [24], it is presented how a relationship is perceived. Relations can be seen as tables of values. These tables have columns not necessarily distinct that consist of "attributes" used to characterize an element that is to be represented by the relation. Each line (formally called "tuple") of this table has values for the attributes. For each column, the values present in every tuple belong to a single domain with a well-defined name, data type, and format. Besides, only atomic values (each value in the domain is indivisible) are allowed.

A schema defines the structure of a relational database. From a schema, tables, their attributes, and relationships between them can be described to be used through a DBMS to create a database. The vast majority of DBMS that implement a relational model uses a standard language to perform queries – the Structured Query Language (SQL) – the relational model is commonly called SQL model. The same extends to the DBMS and databases that implement it.

3.1.2. ACID and BASE Transactional Properties and CAP Theorem

Relational DBMS grants four properties to transactions to maintain data through concurrent access and system failures. These properties are atomicity (A), consistency (C), isolation (I), and durability (D), so they are often referred to as ACID properties. Based on [21,22], a brief description of each of them is presented:

- Atomicity: The transaction must be executed in its entirety or not to be executed. If during the transaction, any failure occurs that prevents the transaction from being completed, any changes that it has performed in the database must be undone;
- Consistency: If a transaction runs entirely from start to finish, without interference from other transactions, it should take the database from one consistent state to another. A consistent database state satisfies the constraints specified in the schema as well as any other database constraints that must be maintained;
- Isolation: the execution of a transaction must not be interfered by any other transaction running at the same time;
- Durability: Changes applied to the database by a committed transaction must persist in the database. These changes must not be lost due to any failure.

A distributed database is defined as "a collection of several logically interrelated databases distributed over a network of computers" [25]. There are three crucial reasons pointed out in the literature for the use of distributed databases: 1) the increase in the volume of data [23], which requires the ability to scale horizontally, that is, to distribute the systems across several nodes. Instead of vertically scaling the hardware – adding more computing resources to the same machine, which would be more expensive and limited; 2) the need to better reflect the distributed organizational structure of companies [26]; and 3) the inherently distributed nature of a range of applications, including industrial ones [26]. There are three ways to implement a distributed database system [22,23]. It is noteworthy that specific systems implement hybrid versions, combining different forms of partitioning [23].

- Single server: There is no distribution. The database runs on a single machine that handles all operations. It is an example of a centralized AAS implementation;
- Partitioning: different pieces of data on different machines. Aggregate models are ideal here, as they form a natural partitioning unit, making certain users access, most of the time, the same server and not need to gather information from different servers, which increases performance compared to server implementation single;

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

- Replication: Data can be replicated in a master-slave schema where the master processes updates to the data and replicates this data to other nodes, or in a peer-to-peer schema where all nodes can process updates and propagate them.

Three other properties are desirable for database systems: availability (A), partition tolerance (P), and consistency (C), which in this case has a slight difference from the concept presented before. The analysis of these properties is fundamental in distributed database systems. The CAP theorem correlates these three properties. According to it, these three properties whose descriptions are presented here cannot coexist simultaneously in a database system:

- Consistency (C): ensuring that all nodes have identical copies of replicated data visible to applications. It is a little different from the consistency concept of ACID properties. In that case, consistency means not violating database restrictions. However, it can be considered that "having the same copy of a data replicated in all nodes that this data is replicated on" is a restriction, so the concepts start to resemble each other;
- Availability (A): each write or read operation will be successfully processed (system available) or will fail (system unavailable). A "down" node is not said to be unavailable;
- Partition Tolerance (P): A partition tolerant system continues to operate if a network fails to connect nodes, resulting in one or more network partitions. In this case, nodes in a partition only communicate with each other.

According to the CAP theorem, only two of the three properties presented can be guaranteed simultaneously. It is worth noting that this choice is not binary, it is possible to relax specific properties so that it is possible to privilege others. However, ACID transactional properties make this flexibility unfeasible. As a kind of alternative, you can have a database that works basically all the time (basically available) and is not consistent all the time (flexible state), only when the writes are propagated to all nodes (eventually consistent) in a distributed system. Thus, the characteristics of this model, named BASE model, although not strictly defined, are presented as:

- Basically available (BA): the system must be available even if partial failures occur;
- Flexible State (S): the system may not have consistent data all the time;
- Eventually consistent (E): consistency will be achieved once all writes are propagated to all nodes.

3.1.3. NoSQL Data Models

NoSQL does not have a solid definition, possibly being better understood as a movement that proposes non-relational database solutions that do not use the SQL language [23,27]. Thus, the term NoSQL (often interpreted as Not Only SQL) used in its technical sense is applied to designate a family of DBMS that have specific characteristics in common (at least for most DBMS), the main one being the non-implementation of relational data model [28]. These features can mean advantages or disadvantages for certain applications:

- They do not implement the relational data model: self-description and the absence of a fixed schema allow for greater flexibility concerning the content stored in DBMS, being suitable for handling semi-structured data [22,28];
- They do not use the SQL language: the absence of a declarative query language, with a wide range of "features" that are sometimes unnecessary, requires more significant effort for developers since the functions and operations have to be implemented through the language of programming [22];
- Absence of ACID transactions: because aggregate-oriented data models generally do not guarantee ACID transactional properties, DBMSs that implement these models have greater efficiency in distributed systems [28,29]. Alternatively, these DBMSs use the BASE model of transactional properties;

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

- Horizontally scalable: The ability of NoSQL DBMS to scale out is linked to two main characteristics: 1) by not having ACID transactional properties (aggregate-oriented models), it allows to relax consistency and thus balance the consistency-latency trade-off in the way which is most suitable for the application, without giving up partition tolerance, as discussed above; 2) the orientation to aggregates allows for a "natural" or intuitive data partitioning unit, as data from an aggregate is commonly accessed together and can be allocated on the same server, which makes the user of this data resort, in the majority sometimes, to the same server [23].

NoSQL DBMS are commonly differentiated based on the way they store the data, that is, the data model employed for the storage. There are four main data models implemented in NoSQL DBMSs. The description of each of these models is presented below:

- Key-value: Key-value DBMSs are possibly the simplest NoSQL systems. These DBMSs store their data in a table without a rigid schema, where each line corresponds to a unique key and a set of self-described objects called value, which can take different formats, from the simplest as strings, passing through tables as in the model relational, reaching more elaborate formats such as JavaScript Object Notation (JSON) and Extensible Markup Language (XML) documents. Thus, they can store structured, semi-structured, and unstructured data in one format (key, value).
  The key-value data model is often represented as a hash table. This data model is aggregate-oriented, meaning that each value associated with a unique key can be understood as an aggregate of objects that can be retrieved in their entirety through the key. The content of these aggregates can be different for each key. The aggregate's opacity guarantees the possibility of storing any data in the aggregates; that is, the DBMS does not interpret the aggregate content, seeing it only as a set of bits that must always be associated with its unique key. This has the practical implication of generally not allowing partial retrievals on aggregate content. The operations implemented by key-value DBMSs are the insertion or update of a pair (key, value), the retrieval of a value from its key, or the deletion of a key;
- Documents: document-oriented DBMSs are those in which data is stored in document format. They can be understood as a key-value DBMS in which the only allowed formats for the values are documents such as XML, JSON, or PDF. A fundamental difference between the document and key-value data models is that the former allows for partial aggregate retrievals as it stores self-described data format. In other words, aggregates are not opaque, they are not seen by the DBMS merely as a set of bits, and it is possible to define indexes on the contents of the aggregates that allow operations to be done on specific items of this dataset.
  As with the key-value data model, the content of each document does not follow a fixed schema. Document labels that guarantee the self-description of the data and enable partial recoveries also allow different keys to having documents with different content (attributes). Thus, it allows the storage of structured and semi-structured data. There are still DBMS that allow the storage of unstructured data such as texts;
- Column Family: In column family databases, data is stored similarly to key-value databases. However, here the value can only be composed of a set of tables, each of which has a name (identifier) and forms a column family. In each of these tables, columns are self-described; they have a key (also called a qualifier) and its value, which are the data itself. Thus, a Column Family database is formed by a table without a rigid structure containing unique keys and a series of Column Family associated with each key in each row.
- Some considerations can be made about this model. The first is that keys do not need to have the same Column Family. The second observation is that, for each key, each column family can only contain the columns of interest; that is, the Column Family do not need to be composed of the same columns for all the keys. The Column

Family form data aggregates that are frequently accessed together, and that because these columns contain their keys, the aggregates are not opaque to the DBMS, being possible to perform partial recoveries through the aggregates through the indexes of the columns;

- Graphs: in graph-oriented DBMS, data is stored in a collection of nodes, which represent entities, and directed vertices, which represent relationships among these entities. The set of nodes and vertices form graphs, in which the two elements that compose them can contain labels and attributes associated with them, which are the data itself. Regarding the characteristics of the data models presented so far, the flexibility in data representation due to the absence of a fixed schema is one of the few similarities between graph databases and the other data models mentioned, as both vertices and nodes can contain attributes different from each other. Concerning differences, the graph model is not aggregate-oriented; usually has ACID transactional properties; it is best suited for single server (non-distribution) implementations; can represent small records with complex relationships to each other; and it is more efficient in identifying patterns, as unlike aggregate-oriented models where partial recoveries can only be made on one aggregate at a time (when allowed), in the graph model they can be done for the graph as a whole.

3.2. Asset Administration Shell

Before presenting the Asset Administration Shell (AAS), it is necessary to introduce the concept of "asset". The IEC describes an asset as "a physical or logical object owned or held in custody by an organization, having a perceived or real value to the organization". Based on this definition, also adopted by the Industry 4.0 Platform [30], it is possible to recognize that an asset can be something physical (a machine, equipment, materials, products) or not (electronic documents, computer programs). Some less intuitive examples of assets are location, time, state of an asset, human beings, and relationships between assets [31]. In summary, the asset is everything that has value and importance for an organization.

It is known that I4.0 characterizes a digital transformation process. For this process to occur, the assets need to be digitized; that is, their data must be taken to the virtual world [32,33]. To perform this mapping in a way that ensures interoperability between systems and components [33], the AAS was created. It corresponds to a standardized digital representation of the asset containing all its technical information and functionalities. The AAS provides a minimum, unique and sufficient description of the asset in its different perspectives relevant to each use case [30,34]. By standardizing the representation format and communication interfaces of assets in the digital world, AAS enables the exchange of information among I4.0 participants, ensuring interoperability between components[30,34]. In summary, the AAS corresponds to the virtual and standardized representation of an asset in the context of I4.0.

The combination of asset and AAS gives rise to Component I4.0 (I4.0C or I4.0 Component). The I4.0C consists of the combination of the physical and real-world, composed by the asset and its respective AAS. The combination of these two elements, with the second "involving" the first, as illustrated in Figure 1, allows services and functionalities to be offered inside (through AAS) and outside (through the asset) of the I4.0C network. These features and services are made possible by the unique identification and communication capability of an I4.0C. Here, it is worth noting that a single I4.0C can be associated with multiple assets depending on the considered granularity. In this way, such a structure can be replicated to different levels of granularity (for example, at different levels of hierarchy). The following subsections discuss details of the AAS structure, elements, metamodel, and implementation perspectives.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

Figure 1. Representation of an I4.0 Component (I4.0C) composed of the asset and the Asset Administration Shell (AAS).

The elements that compose an AAS are divided into classes; each has its attributes used to describe the asset. There are elements in the AAS that can be understood as subclasses, which have the same attributes as a specific superclass from which they are derived but also contain attributes that differentiate them from other elements of the same superclass. The first way to divide the element classes of an AAS is to designate them as Identifiable and Referable. Identifiable Elements are those that have a globally unique identifier. Referable, in turn, has an identifier that is not globally unique, being unique only within the context (defined by an Identifiable) in which it finds itself. The element classes contained within the Identifiable and Referable superclasses are called subclasses and can also, in turn, be superclasses; that is, they can be composed of other subclasses. There is an inheritance relationship of attributes in this hierarchy between classes: subclasses inherit attributes from superclasses.

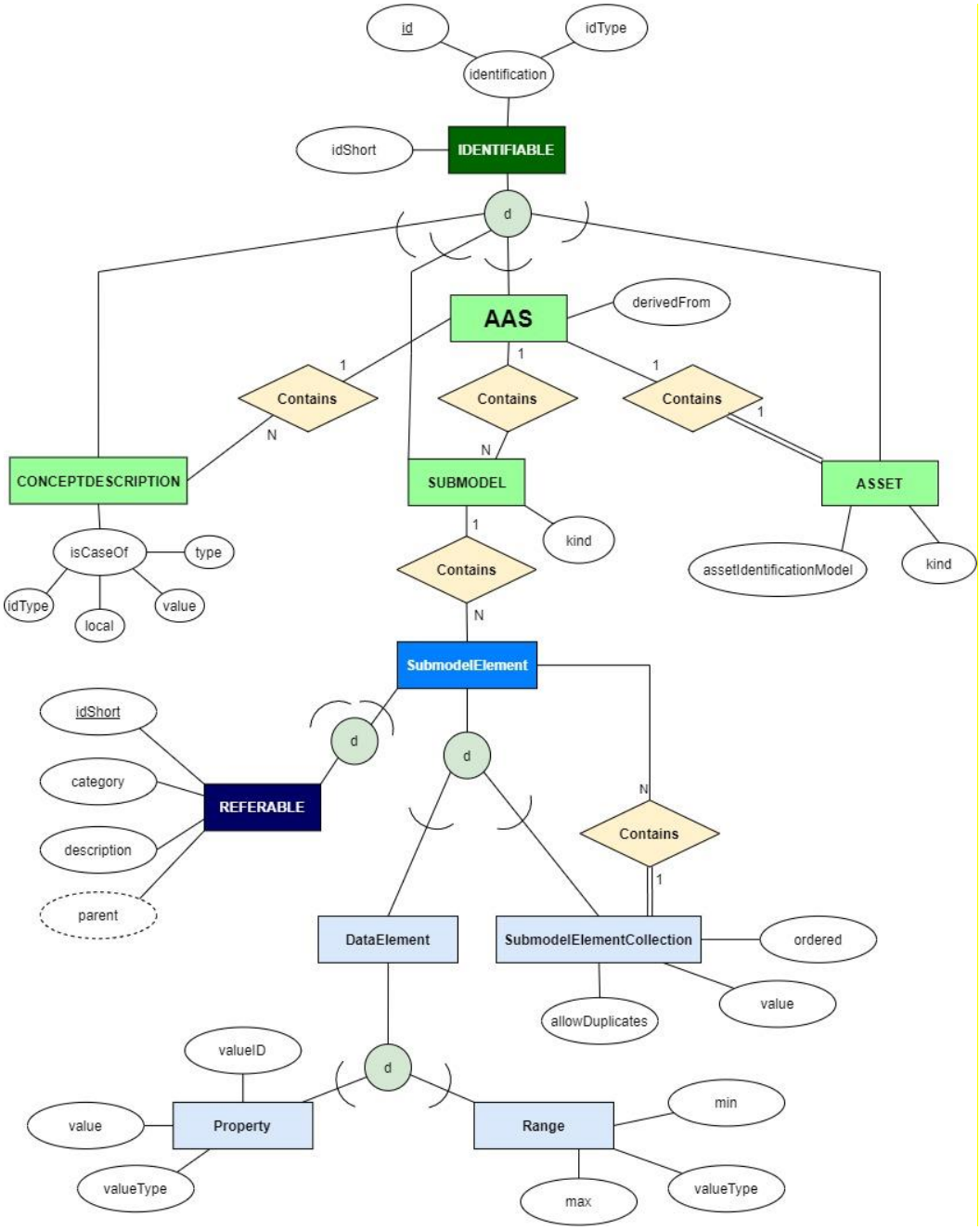
The Identifiable Elements class can present additional domain-specific (owner) identifiers. From the Identifiable class, the "Asset Administration Shell" and "Asset" subclasses have already been described. The subclass "Conceptual Description" defines the standardized semantic description of certain elements. Finally, the subclass "Submodel" allows an asset to be represented in its different perspectives. Each Submodel can describe an asset from an electrical, mechanical, thermal, control, and others point of view.

The "Referable Elements" class has more subclasses than the "Identifiable Elements" class, so only some of them are presented here in more detail. The description of all subclasses can be found in [30]. Among the subclasses of Referable elements, the subclass "Submodel Element" stands out in this work, and it is composed of elements suitable for the description and differentiation of assets in a perspective specified by the Submodel. This class of "Submodel Elements" can be understood as a superclass, in which some of the main subclasses are "Submodel Element Collection" - a collection that can be composed of all other classes with the same hierarchical level - and "Data Element". Data Elements, in turn, form another superclass with one of the important AAS element subclasses, the "Property", described in more detail in the next paragraph.

The "Properties" class contains elements that allow representing the characteristics of an asset, given a perspective defined by the Submodel in which they are found. These elements are standardized by the IEC 61360 and can be found in the IEC Common Data Dictionary (CDD, common data dictionary) or eCl@ss repositories [30,35,36]. In the IEC CDD repository, a property has, in addition to its value itself, some additional data such as code, version, and revision, identifier, and definition. The free digital version of the IEC CDD provides examples of properties for some specific domains. The complete list of AAS element classes, including those that do not qualify as Identifiable or Referable, can be found in [30].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

Once the structure and some of the main components of the AAS are presented, it is possible to illustrate its metamodel. Figure 2 illustrates this metamodel with the components that were presented through an extended entity-relationship schema. In this schema, the entities (rectangles) represent the AAS element classes. Ellipses contain attributes of these elements. Some of these attributes are of the composite type; that is, they are sets of other attributes. Ellipses with a dashed outline indicate derived attributes, that is, attributes whose value is derived from other elements in the schema. Attributes whose name is underlined consist of identifiers ("id" for Identifiable Elements and "idShort" for Referable Elements). Diamonds indicate relationships between AAS elements. Most elements are of the "Contains" type, which indicates that one or more elements are a composite of others. It is observed that one of the diamonds corresponds to a specific element of the AAS that is not mapped as an entity but as a relationship – it is the "Bill of Materials". Circles with a "d" in the center indicate a specialization; they serve to indicate the disjoint subclasses that constitute a superclass. In the example shown, it can be seen that the superclass "Referable" is composed of the subclasses "AAS", "Asset", "Concept Description", and "Submodel".



Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

Figure 2. Extended-Entity-Relationship schema represents the metamodel of the AAS with a few elements.

The standard solutions proposed for I4.0 need to be comprehensive enough not to limit the possibility of carrying out the Smart Factory in most different and organizations. In this sense, no specific strategy for implementing the AAS is imposed by standardizing the digital format of representation and exchange of information. In [37], some of these possibilities are explored, taking into account different implementation perspectives. The different possibilities provide characteristics that can be advantages or disadvantages for specific applications. These characteristics include computational power, availability, performance and latency, security and reliability, maintenance, administration and management cost, failure identification, and recovery. Here, three perspectives of AAS implementation presented in [37] are described, along with the advantages and disadvantages.

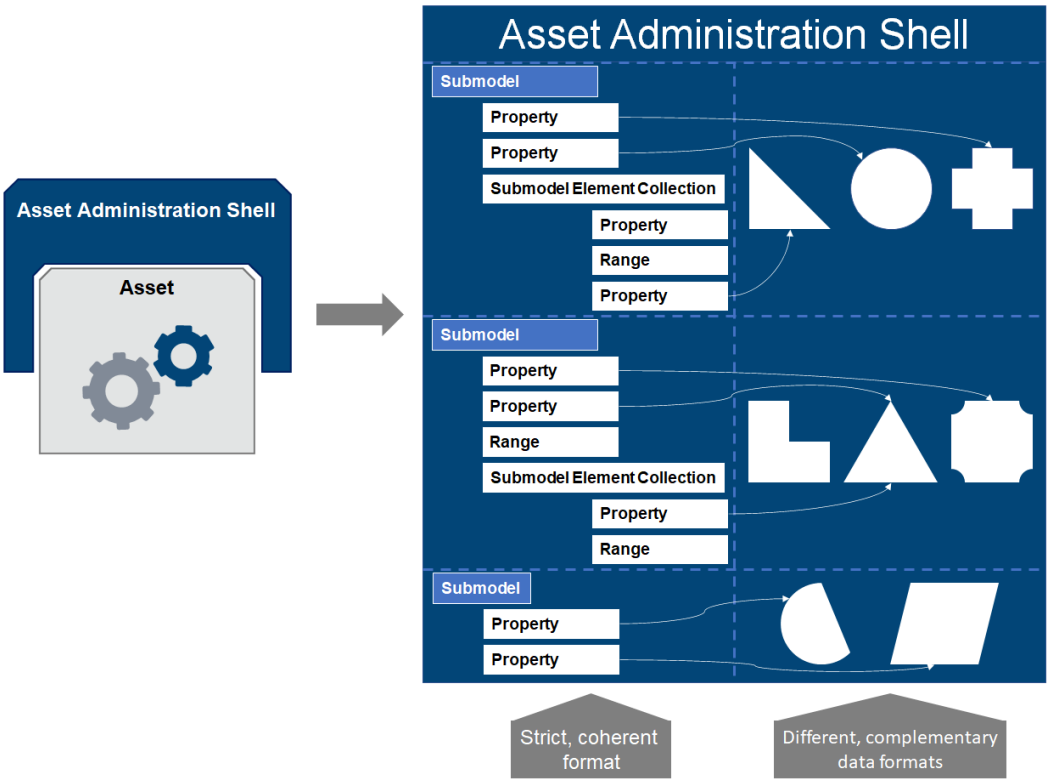


Figure 3. Schematic representation of an example of AAS. Adapted from [35].

The first issue to be discussed regarding AAS implementation is the computing platform. Three possibilities of implementation are presented, as illustrated in Figure 3. In the first one (Figure 4a), the AAS is embedded in the asset, which in turn contains an execution environment for its digital representation. It is the case that an implementation based on edge computing platform can be used as an example for such an implementation. In a second possibility (Figure 4b), the AAS can be physically separate from the asset but residing in the local IT infrastructure, connected to the asset through a local network. This case corresponds to a fog computing platform-based implementation. As a third possibility (Figure 4c), the location of the AAS can be even further away from the asset in a cloud computing platform-based implementation. In this case, AAS and asset connect via external internet networks.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

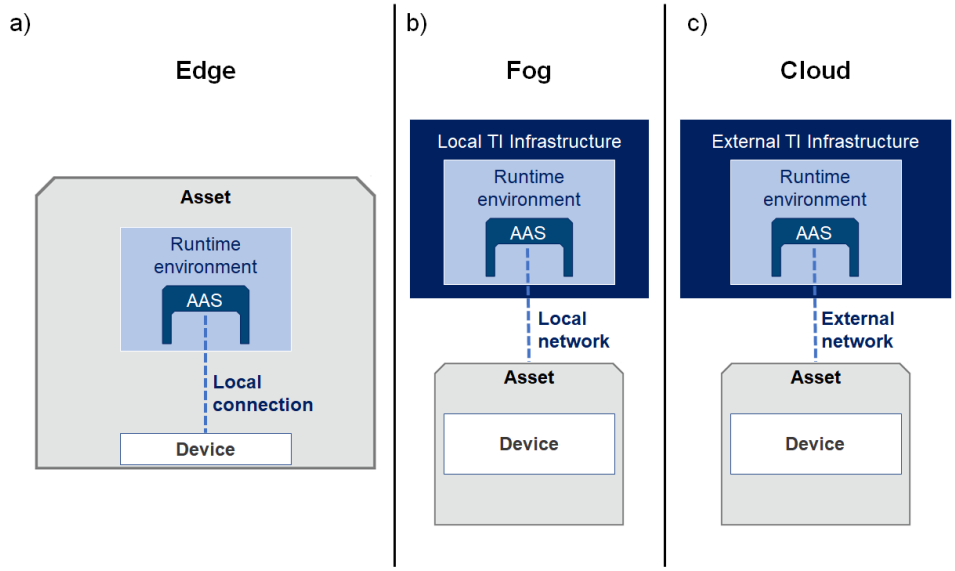


Figure 4. Computing platform perspective for implementation of the AAS.

The second implementation perspective concerns the scalability of AAS. In a simplified way, scalability is related to the possibility of distributing data storage and processing across multiple nodes of a network. In this subsection, three possibilities for AAS distribution are considered, as illustrated in Figure 5: (a) centralized, in which all information and services are allocated in a single node; (b) loosely coupled distributed, where different nodes store information for the same asset (same identifier) and can be accessed individually; and (c) distributed with aggregator node, which differs from the previous implementation by including an aggregator node, which gathers information from the nodes on which the AAS is distributed, forming a single data access point.

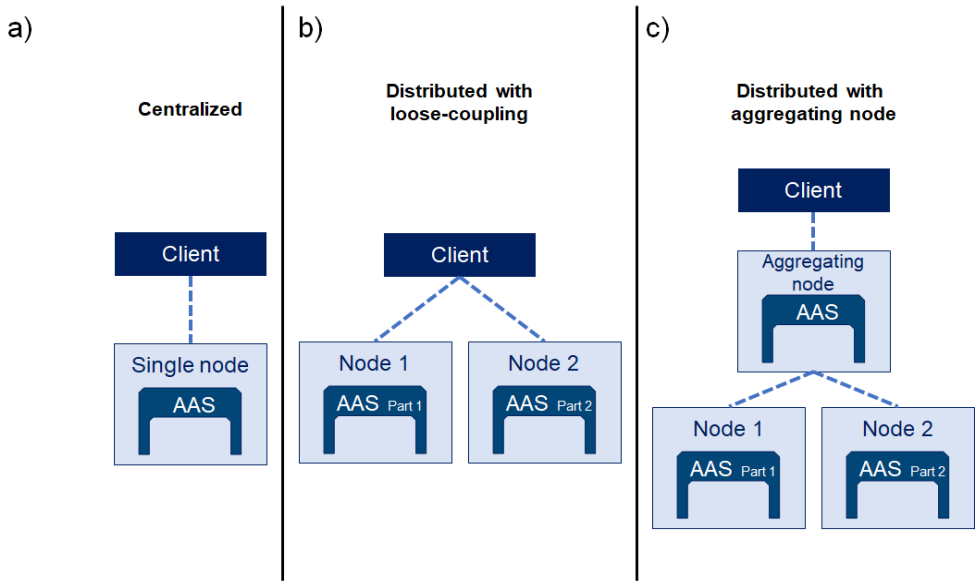


Figure 5. Distribution perspective for implementation of the AAS. Adapted from [37].

In Figure 4, the implementations differ in the distance between the AAS execution environment and the asset. However, no further consideration is made concerning the specific execution environment, which defines a form of AAS virtualization. This form of virtualization is an implementation perspective that implies advantages and disadvantages for the application. Three possibilities are presented and illustrated in Figure 6:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

the implementation based on (a) operating system, (b) hypervisor, and (c) container. In the first case, the operating system is the AAS execution environment itself; that is, the execution environment consists of a process of a dedicated operating system or running inside another process. In the second case, the AAS execution environment is a virtual machine (VM). Multiple virtual machines with their own operating systems are allocated to a host (host) machine; they use its hardware, and a hypervisor manages it. As in the previous case, AAS would still function as a complete operating system process, but in this case, this process would share hardware resources with other operating systems and applications. Finally, in the third possibility, the AAS execution environment consists of containers, which run on top of an operating system. Unlike virtual machines, applications run in containers are run on the host machine's operating system, requiring only minimal resources such as applications and APIs needed to run the application, in this case, the AAS [38]. Two issues are related to the AAS execution environment, namely, to its virtualization form: isolation and performance. These two characteristics form a trade-off, as pointed out by [39].

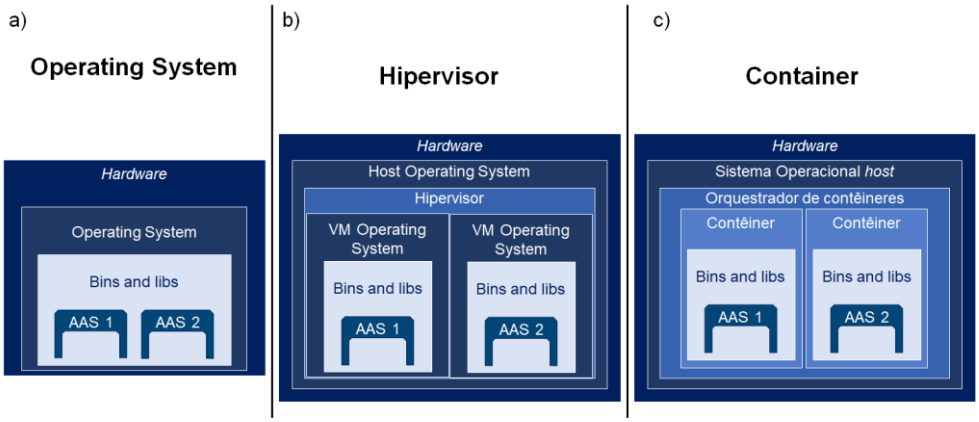


Figure 6. Virtualization perspective for implementation of the AAS.

4. Results

This section presents results regarding the importance of technologies and methods gathered under the term "Big Data & Analytics" for I4.0 and correlates the characteristics of the data in this context with those of the SQL and NoSQL databases.

4.1. Data Characteristics in the Context of I4.0

There is still no consensus on the technological pillars that support Industry 4.0 (I4.0). It is also observed that most authors put "Industry 4.0" and "Fourth Industrial Revolution" as synonyms, considering that there is no distinction between the phenomena, making it even more challenging to identify the technological pillars associated with each of the concepts. Table 2 presents the views of different authors about the key enabling technologies of the I4.0.

Table 2. Key enabling technologies for Industry 4.0 and/or the Fourth Industrial Revolution, according to different authors.

Technology	**Rüssmann (2015)** [40]	**Bechtold *et al.* (2014)** [41]	**Lichtblau *et al.* (2015)** [42]	**Bauer *et al.* (2015)** [43]	**Petrillo *et al.* (2018)** [44]	**Wan, Cai, Zhou (2015)** [45]
Big Data & Analytics	x	x	x	x	x	x
Advanced robotics	x	x		x	x	
Systems integration	x				x	x

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

Internet of things	x		x	x	x	x
Simulation	x				x	
Additive manufacturing	x	x	x	x	x	
Cloud computing	x	x	x		x	x
Virtual/augmented reality	x		x	x	x	
Cybersecurity	x				x	
Machine-to-machine		x		x		
Mobile technologies		x	x			
Location and detection technologies			x			
Human-machine interfaces			x	x		
Authentication and fraud detection			x			
Smart sensors			x			
Interaction with customers			x			
Community platforms		x				
Embedded projects						x
Self-guided vehicles						x
Social networks						x

Despite this lack of consensus mentioned, it can be observed that there are certain convergences between authors and organizations about what would be the enabling technologies of I4.0. It can be seen from Table 2 that the only technology identified as such in all the works consulted was Big Data & Analytics. Thus, although there is no total consensus among the authors, the relevance of Big Data & Analytics for the industry in the coming decades can be recognized. In brief, the term Big Data & Analytics comprises i) datasets "characterized by a high volume, velocity and variety to require specific technology and analytical methods for its transformation into value" [46]; ii) as well as the technologies and data analysis methods (big data analytics) themselves for this type of information asset, that is, specific to its characteristics [42,47].

Data generated, collected, transmitted, and possibly analyzed in real-time will be part of the Intelligent Factory [48]. The fact that Big Data & Analytics systems are considered a pillar of I4.0 comes from the possibilities of improvements that can occur in a company based on its available data. [13,48] state that this type of system can be used to increase productivity; better risk management; cost reduction; and aid in general decision-making. For this reason, this set of technologies is considered fundamental for I4.0 [49]. Considering that Big Data & Analytics is a key enabling technology of I4.0, the five dimensions of big data presented can be used to obtain a more general description of the nature of data in the context of I4.0. Still based on the definition of big data presented, four dimensions characterize this information asset: the volume, speed, variety, and value of data. A fifth dimension – veracity – is still considered by some authors to encompass the reliability of the data [48,50]. These five dimensions, also called "5V", characterize big data and have direct implications for how data is stored, manipulated, and analyzed [51]; that is, they require technological solutions and specific methods for these functions. A description of these five dimensions of big data is presented with a discus-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

sion of how they are manifested in I4.0, i.e., the concept, structure, and implementation perspectives of the Asset Administration Shell.

- Volume: is associated with the amount of data involved in big data applications. Although there is no consensus on a reference value in terms of data volume for a database to be characterized as big data, this dimension is usually characterized by petabyte (PB) or exabyte (EB). According to [48], data volume in the modern industrial sector tends to grow by more than 1000 EB per year.
  In the context of I4.0, the digitization of the most diverse assets and the communication between them leads to an "unprecedented growth" in the volume of data, according to [52]. [45] state that big data cannot be manipulated in a single computer, requiring a distributed architecture. Thus, it is noted that the concern with the volume of data in I4.0 is reflected in the perspectives of implementing the AAS concerning its distribution. It can be argued that centralized implementations are suitable for smaller volumes of data, while distributed implementations are suitable for large volumes.
- Variety: refers to the different formats that the data can contain. Big data applications can involve structured data, such as rigidly structured tables populated with scope-limited values; semi-structured as documents with a pre-defined template; and unstructured, such as multimedia content (image, audio) [53]. It is possible to argue that such heterogeneity may exist in the industrial context, but it is possibly more "controlled". Despite this fact, variety is still a characteristic of the data in I4.0, considering that the AAS proposal presupposes a standardized format of representation and exchange that must be able to include assets of the most diverse natures. Thus, variety is a feature of the data in I4.0.
- Velocity: This dimension can be understood as having two components – the speed with which data is collected and the speed with which it is processed [53]. In older big data applications, processing was commonly done in batches, so the speed at which data is captured is critical to ensuring its reliability. Newer applications enable data processing in real-time and in data streams so that, in addition to ensuring data reliability, the capture speed must be consistent with the data processing speed [48,50,53].
  In addition to the high growth rate of data volume, which has already been mentioned and is more associated with data capture speed, one can also discuss data processing speed and data analysis in I4.0. Batch processing consists of processing a large volume of data at a time. The literature reports examples of this type of processing in an industrial environment [54,55]. Furthermore, Data Warehouse systems are typical examples of this way of processing and analysis. Applications of real-time processing and analysis in an industrial environment are also reported in the literature [56–58]. Comparing the AAS implementation platforms - edge, fog, and cloud - the last two are more suitable for batch processing since, in general, they have greater computational capacity than the first. However, this finding can be changed with the evolution of technology and the possibility of increasing data processing and storage capacity in devices closer and closer to the edges of networks. The asset-based implementation, that is, on edge devices, favors real-time processing due to low response latency.
- Value: is associated with the value that can be extracted from the data through data analytics. Extracting value from data consists of converting the data into entities with a higher hierarchical level [53]. It involves a series of data analysis techniques, including machine learning, that requires a multidisciplinary approach; and, above all, it receives the name "value" because it offers prospects for improvement and cost reduction in terms of products and processes in the industry [48,59] so that it is possible to argue that there is a loss in not extracting value from the data. Extracting value from data in a significant data context presupposes the application of specific technologies and analytical methods. This dimension highlights the importance of data for the industrial sector, as it brings the possibility of implementing improve-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

ments in organizations based on data.

Extracting value from data involves employing data analysis techniques in a multidisciplinary approach, which can translate into a naturally distributed organizational structure of a company or institution. Regarding the AAS implementation perspectives, its distribution in different network nodes, in fact, better reflects the structure of organizations today [37]. In these situations, each of the subdivisions of an organization is responsible for a fraction of the AAS or the whole AAS that concerns it. Because applying data analysis techniques in an organization to extract value from them requires the integration of different perspectives, it assumes that distributing AAS across different nodes also requires that these "AAS fragments" (or different AAS) be integrated to extract value from your data. Taking up the perspectives of AAS implementation regarding its forms of distribution, the distributed solution with aggregator node can be an adequate solution for data integration. This does not mean that the aggregator node needs to act as a master node of the network, which manages all routine transactions, but that it can act as a data integration node and as a source of access by those members of the organization who intend to extract data value.

- Veracity: is associated with the reliability of the captured data. [60] argues that the veracity dimension has three components: objectivity/subjectivity, which is more linked to the nature of the data source; deception, which refers to intentional errors in the content of the data or malicious modifications thereof; and implausibility (implausibility, irrationality) of the data, which refers to the quality of the data in terms of its validity, that is, its degree of confidence. Such concern is observed in the context of I4.0 as authors consider cybersecurity as a pillar of I4.0 (see Table 2), which presupposes protection against errors and intentional modifications to the data. It is also observed in the AAS implementation perspectives, where the virtualization strategy directly affects the isolation between applications [61] and, consequently, confidentiality and data integrity.

4.2. Data Models in the Context of Industry 4.0

The results of the systematic review whose procedure was detailed in Section 2 demonstrates the gap that exists in the proposals for system architectures for I4.0 in terms of database design. Of the 139 papers resulting from the application of the search string on the Google Scholar platform, which reports the implementation of the AAS, 25 of them scored higher than five based on the criteria adopted and were analyzed. Only 32% (8 papers) at least inform the data model, or DBMS used, suggesting that studies in this area discuss this issue. However, databases should not be understood as mere tools for data storage but as essential components of architectures, which can impact their performance [62]. For this reason, database solution choices should not be made arbitrarily but based on criteria, application characteristics, users, and data.

Given the reality exposed in the previous systematic review, i.e., the gaps in the implementation of database solutions in the context of I4.0, it can be introduced a correlation between the data characteristics described in the previous section – considering the five dimensions of big data – and the characteristics of relational and NoSQL data models, discussing the adequacy of these models to the context of I4.0.

4.2.1. Volume

The need for database distribution is a characteristic that imposes limitations on the use of the relational model for large volumes of data [63,64]. The main problem associated with the use of relational databases to implement distributed database systems is linked to the CAP theorem and, consequently, to ACID transactional properties. A single server system is a CA system: there is no partition tolerance because there is no partition on a single machine. Therefore, the two other properties – availability and consistency – are guaranteed. Most relational database systems are CA, and licenses for this type of DBMS are generally marketed to run on a single server [23].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

On distributed systems, there is the possibility of partitioning the network. In this type of system, it is only possible to leave off partitioning tolerance if, in the event of a network partition failure, the system becomes completely inoperative, which is critically undesirable in some instances. Thus, in distributed systems, it is generally not desirable to leave off the tolerance of network partitioning; that is, it is not desirable to have a distributed CA system. The other possibilities are leaving off consistency or availability. Therefore, the essence of the CAP theorem can be understood as: in a system that may be subject to partitioning, one must prioritize between consistency or availability. This turns out to be, in fact, a trade-off between consistency and latency: to have consistent transactions, a certain amount of time is needed for data changes to propagate to all copies, and the system can be available again [29]. In summary, there are relational databases that can be horizontally scalable; that is, they can be distributed [65], but the possible high unavailability of the system can make this distribution unfeasible.

Because transactions that adopt the ACID model are strongly consistent, it is not possible to balance the trade-off between consistency and latency in distributed databases that use this guarantee model [29]. For this reason, maintaining ACID properties generally implies in higher latency [23,66] in a distributed database that implements the relational model. For high availability, data needs to be replicated across one or more nodes, so if one node fails, the data is available on another. Replication can increase availability and performance by reducing the overhead on nodes for reading operations. However, for "write" operations where one want to ensure that all nodes have an up-to-date copy of the data, one can experience a loss of performance (one must wait until the data is replicated across all nodes). On the other hand, in systems that adopt the BASE model, lower latency can be achieved, but inconsistencies can occur during a specific time interval (inconsistency window) since the different nodes can present different versions of the same record. Thus, it is understood that NoSQL systems, especially those oriented to aggregates, are beneficial for I4.0 in distributed database implementation scenarios, which usually contain large volumes of data, as they facilitate horizontal scalability.

In addition to performance, aggregate orientation is another reason why NoSQL data models better suit distributed database systems. Specific applications may contain data sets that are frequently accessed and manipulated together. In distributed systems, these sets can form a natural distribution unit [23] so that interested users are always directed to one or more specific nodes where they are stored. In databases, these sets are called aggregates: a rich structure formed by a set of data (objects) that can be stored as a unit, as they are often manipulated in this way. Elements of AAS as Submodel (set of Submodel Elements), Submodel Element Collection, and AAS itself (set of other elements) can be understood as aggregates. As such, aggregate-oriented NoSQL models are helpful for distributed implementations of AAS.

In terms of the implementation data model, the main problem of the relational model regarding the representation of aggregates is its rigid structure, which makes it impossible to treat data sets as a unit. The relational model allows representing the entities and relationships that are part of an aggregate. However, it does not allow to represent the aggregate itself; that is, it does not allow identifying which relationships constitute an aggregate nor the boundaries of this aggregate. For applications looking to process aggregates as a whole, the NoSQL key-value data model is beneficial because the aggregate is opaque. In case it is necessary to access parts of an aggregate, the document data model is more suitable than the key-value, as the aggregates are transparent in the case of the first; that is, partial operations can be performed on the data of the aggregates. For processing data from simpler formats like numeric values, strings, etc., Column Family are also suitable for the purpose.

4.2.2. Variety

This dimension manifests in scenarios where data heterogeneity is present. Such scenarios may require flexibility in databases. A flexible data structure is not observed in

 doi:10.20944/preprints202111.0019.v1

the relational model, both in terms of the relationship scheme and the restrictions imposed by domains on possible values for attributes in the relationships. In the context of I4.0, this inability (or difficulty) can be verified in the attempt to create an AAS metamodel in a relational schema. Since the AAS must be able to contemplate all I4.0 assets, it has a vast number of classes (entities) that represent each of its elements what would be translated into a large number of relations which could be even more significant if normalization procedures are applied. Associated with this complexity arising from the mapping of the AAS metamodel in the relational scheme, assets also have heterogeneity. When building a base composed of different assets, this heterogeneity can imply many null fields, which is undesirable.

The flexibility of NoSQL systems makes them suitable for I4.0. Such flexibility enables the storage of semi-structured data, which best characterizes AAS. Technical reports from the Industry 4.0 Platform [30] and academic papers [67–69] present AAS implementations in XML and JSON format, which suggests that document-oriented NoSQL systems are advantageous, although it is not the only one capable of storing semi-structured data. This document encoding type is supported by essential communication technologies relevant to the I4.0, such as OPC UA [70] and HTTP. In summary, NoSQL data models adapt to the characteristic variety of data in I4.0, enabling the storage of heterogeneous records in the same DB. Thus, the flexibility of NoSQL models has its importance in the context of I4.0.

4.2.3. Velocity

This dimension has two components: velocity of data collection and data processing. In terms of database systems, the "processing velocity" component has significant implications for database design. As described in the previous section, in older Big Data & Analytics applications, data processing often took place in batches, offline, while newer applications adopt real-time processing and stream processing. These two forms of processing are essential to guide the choice of database.

Before introducing the discussion of data models suitable for batch and real-time processing, it is essential to introduce the speed, consistency, and volume principle, or SCV principle (speed, consistency, and volume, respectively). While the CAP theorem presented in Section 3.1.2 concerns data storage, the SCV principle deals with data processing. The first attests that it is impossible to simultaneously guarantee consistency, availability, and tolerance to the partition. The second states that it is impossible to guarantee processing speed, consistency of results, and processing large volumes simultaneously. Based on [71], each of the elements that compose the SCV principle is described:

- Speed: deals with the speed at which data can be processed. To calculate the processing speed, the time spent to capture data should not be taken into account, considering only the actual processing time;
- Consistency: concerns the accuracy and precision of the results obtained from data processing. Inconsistent systems cannot use all available data to be processed, adopting sampling techniques, which leads to less precision and accuracy of results. On the other hand, systems with greater consistency use all available data in processing, obtaining more precise and accurate results;
- Volume: deals with the amount of data that can be processed. Large volumes of data require distributed processing, while smaller volumes can be processed centrally.

Dealing with large volumes of data often requires allocating both storage and processing. For this reason, from the point of view of the SCV principle, there is a trade-off between processing speed and consistency of results, whereas, according to the CAP theorem, there is a trade-off between availability and consistency. To identify suitable ways to deal with these trade-offs, it is possible to consider two essential characteristics of batch processing: the intention to provide complex and valuable data analysis so that the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

precision and accuracy of the results are necessary; and the requirement that multiple records be accessed at one time.

The two characteristics mentioned about batch processing can guide the choice of the most suitable data models for this type of application. Taking into account the first characteristic, the importance of ensuring data consistency is noted, both from the CAP theorem's point of view and the SCV principle. Thus, relaxation of consistency may be undesirable in batch processing. As the ACID transactional properties generally implemented in relational and graph-oriented DBMSs do not allow for consistency relaxation, these systems are suitable choices for batch processing. Furthermore, it is known that aggregate-oriented NoSQL systems cannot efficiently meet the second feature, as they can only access one aggregate at a time. This would imply an even lower processing speed for large volumes of data if the operations to be performed require information from different aggregates. Thus, although transactional properties imply a larger downtime window, this loss of efficiency can be, in a way, offset by greater efficiency in accessing multiple records.

Still regarding the dimension “processing speed” in big data, the case of real-time processing is now analyzed. Also, taking into account that significant data volume cannot be left off, which imposes the need for storage and processing distribution, there are the same trade-offs presented above, between processing speed and consistency of results (SCV principle) and between availability and consistency (CAP theorem). However, the real-time processing requirements are different. In this case, as delays are unwanted, trade-offs tend to prioritize speed and availability over consistency. For this reason, the implementation of AAS based on edge computing platform is suitable for this type of processing, as being closer (or even embedded) to the asset, delays tend to be smaller.

It is known that ACID transactional properties do not allow the relaxation of consistency in favor of increased availability. Thus, aggregate-oriented NoSQL systems, which implement the BASE model of transactional properties, may be more suitable solutions for real-time processing. However, the level of consistency required by the application must be taken into account so that, by maximizing availability and processing speed, precision and accuracy requirements are not violated. Aggregate-oriented models are even more efficient; they guarantee higher processing speed if they do not need to perform operations on multiple aggregates simultaneously.

4.2.4. Veracity

Veracity concerns the reliability of the data. In Section 4.1, the three components of veracity were presented: i) the objectivity of the data, which fundamentally depends on the nature of the data source; ii) deception, a problem in the field of cybersecurity; iii) and the implausibility. Of these dimensions, [53] points out some causes for veracity problems associated with implausibilities, such as inconsistency, latency, and incompleteness. Therefore, it is observed that these causes and, consequently, the veracity is essential for the database design.

It is possible to associate the causes of implausibility with the properties of the CAP theorem and thus discuss the “truthfulness” dimension for different database systems. The inconsistency that affects the veracity of the data is directly linked to the consistency referred to in the CAP theorem. Latency is associated with the availability property, as seen in 3.1.2. The issue of incompleteness, in turn, is not directly associated with a property of the CAP theorem but with the transactional guarantee of atomicity, which establishes that a transaction must be performed entirely or not be perfomed at all. Thus, there is a foundation to discuss the impact of data models on veracity.

ACID transactional properties contribute to data veracity by enabling consistency and atomicity to operations. However, such properties imply high unavailability, which translates into delays in operations. Returning to the "speed" dimension of big data, if the data processing speed obtained through a system with ACID properties is consistent with the speed of data entry into the system, so there is no processing of outdated data,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

then these database systems can be employed. Relational and graph-oriented DBMS generally adopt such properties.

Adopting the BASE model of transactional properties promotes an increase in availability at the expense of relaxation of consistency, which implies a decrease in the delay at the price of the possibility of occurrence of inconsistencies. This does not mean that the BASE model is necessarily an inadequate choice when one wants to guarantee veracity based on completeness and consistency. The BASE model does not make it impossible to guarantee consistency but allows a balance of the trade-off between consistency and availability to suit the application's needs better. Thus, one of the properties can be prioritized according to the characteristics of the application and the problems related to implausibility, whose susceptibility to the occurrence is greater. Thus, aggregate models can guarantee the veracity, dealing with delay, incompleteness and inconsistency not simultaneously but balancing the problems according to the application demand.

This discussion is enriched with the introduction of a way to classify databases is by dividing them into integration and application databases. The former store's data from multiple applications in a single database. This type of system has a much more complex structure than would be required by individual applications, as there is a need to coordinate and orchestrate applications, which differ above all in terms of performance requirements in terms of their operations. Application databases, in turn, are accessed and updated by a single application. This type of implementation allows databases to be encapsulated to applications, and the integration between them occurs through services so that application databases are fundamental for web applications and service-oriented architectures in general [23]. In an I4.0 context, it is possible to observe that the AAS implementation perspectives regarding its virtualization support both types of databases, especially concerning the degree of isolation between applications.

Integration databases generally implement the relational model, as the ACID properties confer precisely the desired concurrency control to coordinate the requests of different users/applications of the database [72]. However, for application databases, the relational model entails specific unnecessary and even undesirable characteristics: an application database usually requires a much smaller number of operations offered by the SQL language [23]; and ACID transactional properties which ensures concurrency control becomes unnecessary, as only one application accesses the database [23].

4.2.5. Value

It deals with the use of data to implement improvements in the organization, whether in processes, products, services, among others. Although the extraction of value from the data is more linked to data analysis than to its storage, here is an excerpt of this dimension regarding database management systems, considering the interdisciplinarity and distribution of data in organizations. Thus, the analysis of the adequacy of database systems for this “value” perspective is mainly done taking into account the importance of AAS distribution and the need for integration to extract value from asset data.

Network nodes that only contain AAS data referring to an organizational unit of the institution are those that process more routine transactions, the so-called On-line Transaction Processing. The databases of these nodes are called operational or transactional databases. A network node that promotes data integration, in turn, processes transactions with an analytical purpose, On-Line Analytical Processing, and provides data for algorithms and other subsystems, acting in fact as an integrator database. In an institution with a distributed organizational structure, the data models that enable the horizontal scalability of the database, that is, the NoSQL DBMSs oriented to aggregates, are suitable for implementing "transactional" nodes, that is, those that process routine data transactions specific AAS that pertain to a unit of the organization. If AAS distribution is not done through the DBMS itself but through application databases that communicate through service interfaces, then NoSQL data models are still applicable. The flexibility

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

provided by these models allows each department to adopt the data models that best suit their application.

An integrator node is usually built from the so-called multidimensional data model at the conceptual level of data abstraction [73,74]. This is where the value is extracted from the data utilizing (big) data analytics methods. The multidimensional model is generally mapped at the implementation level through a relational scheme, although there are literature works that seek to map the multidimensional model in NoSQL models [75–77]. In particular, the importance of this mapping for the graph model is highlighted: the integrator node usually performs processing in batches, and, based on the discussions in the previous subsection, it was argued that the graph-oriented model is suitable for this type of processing.

5. Discussion

Big data dimensions and other data characteristics in I4.0 were addressed in the previous subsections to discuss the suitability of data models to different realities of I4.0. However, the database design requires that interrelationships among these characteristics be analyzed, as it is possible to observe that one dimension can affect the others regarding the data model to be used. The dimensions "volume" and "velocity", for example, are correlated according to the SCV principle. When dealing with the "veracity" dimension, the impact of the BASE and ACID models on the veracity of the data was discussed. However, the use of one or another model of transactional properties also affects the distribution of data associated with the "value" dimension. The variety dimension, which concerns the possibility of storing data with more complex structures and, therefore, presupposes the use of more flexible data models such as those oriented to aggregates, for example, also implies the speed of processing multiple records, which is inferior in this type of data model.

Two qualitative analyzes are presented to synthesize the discussions presented so far. The first of them is represented in Table 3, in which the dimensions "volume", "velocity", and "veracity" are associated with the two models of transactional properties, that is, BASE and ACID. As seen earlier, the first model is generally implemented in aggregate-oriented NoSQL databases, while the second is implemented in relational and graph databases.

Table 3. Most suitable model of transactional properties according to the volume, velocity, and veracity of data.

Volume	Velocity	Veracity	Suitable model of transactional properties
Low	Low	Low	BASE
		High	Both
	High	Low	BASE
		High	ACID
High	Low	Low	BASE
		High	ACID
	High	Low	BASE
		High	None

Table 3 does not refer to one or more data models specific to each scenario. The "variety" dimension, in addition to data linkage complexity and the flexibility of access, can be taken into account so that, based on a model of transactional properties, the choice for a data model can be made. Tables 4, inspired in [51], synthesizes these three characteristics also qualitatively, suggesting the most appropriate data model with ACID transactional properties. Likewise, Table 5 suggests the most suitable data models with

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2021 doi:10.20944/preprints202111.0019.v1

BASE transactional properties according to the veracity dimension, access flexibility, and data linkage complexity.

Table 4. Most suitable data model with ACID properties according to veracity, access flexibility, and data linkage complexity.

Variety	Access flexibility	Data linkage complexity	Suitable logical data model
Low	Low	Low	Relational
		High	Graph
	High	Low	Relational
		High	Graph
High	Low	Low	Graph
		High	Graph
	High	Low	Graph
		High	Graph

Table 5. Most suitable data model with BASE properties according to veracity, access flexibility, and data linkage complexity.

Variety	Access flexibility	Data linkage complexity	Suitable logical data model
Low	Low	Low	Key-value
		High	Column Family
	High	Low	Document
		High	Document
High	Low	Low	Column Family
		High	Document
	High	Low	Document
		High	Document

It is possible to observe that, when considering the specifications of a given application along the dimensions, the choice for a database generates trade-offs in terms of the requirements that can be met. In specific applications, conflicting characteristics from a database standpoint can be equally important. For this reason, it is common to find applications, especially in service-oriented architectures, in which multiple databases are used to meet the different application specifications satisfactorily. Works in this area are referred to as polyglot persistence [23,62], in which each database is responsible for managing data from a part of the application.

6. Conclusions

This work presents different contributions regarding the database in an Industry 4.0 (I4.0) context. Given the importance of understanding the characteristics of the data for the design of a database, the article provided a comprehensive description of the data in the context in question, identifying, for this purpose, the technologies, methods, and standards related to data that are show fundamentals for the I4.0.

Regarding the database design itself, this paper seeks to corroborate the assertion that, among the works that propose system architectures for I4.0, including adopting its standardizations such as Asset Administration Shell (AAS), few demonstrate evident concern and justification about the choice of data models to be used and how databases can influence the performance of these architectures. Subsequently, based on the characterization of the data in an I4.0 context, analyzes were made of how the characteristics of SQL and NoSQL data models fit into the five dimensions of the data - volume, velocity,

variety, veracity, and value. These analyses were summarized in Tables 3, 4, and 5, in which hypothetical scenarios were built based on these five dimensions of data and other characteristics such as flexibility of access and complexity of data connections. The transactional guarantee models (ACID and BASE) and the SQL and NoSQL data models that best fit were suggested for each scenario.

The results presented in this paper adopted a qualitative comparison between data models. Works found in the literature propose comparisons between the performance of SQL and NoSQL databases based on quantitative metrics [28,78,79]. Future work can explore the dimensions by which the data were characterized in this paper and quantitatively assess the performance of the data models for the scenarios presented.

Supplementary Materials: not applicable

Author Contributions: For research papers with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used "Conceptualization, V.F.O. and F.J.; data curation, V.F.O. and F.J.; formal analysis, V.F.O. and P.E.M.; funding acquisition, F.J., M.A.O.P., and P.E.M.; investigation, V.F.O. anfd F.J.; methodology, V.F.O., F.J. and P.E.M.; project administration, V.F.O., F.J., M.A.O.P., and P.E.M; resources, V.F.O., F.J., M.A.O.P. and P.E.M; software, V.F.O.; supervision, F.J., M.A.O.P., and P.E.M; validation, V.F.O., F.J., M.A.O.P.; and P.E.M; visualization, V.F.O., F.J., M.A.O.P. and P.E.M; writing—original draft preparation, V.F.O and F.J.; writing—review and editing, V.F.O., F.J., M.A.O.P., and P.E.M. All authors have read and agreed to the published version of the manuscript."

Funding: This research was supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), grant number 88887.508600/2020-00; Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP, São Paulo Research Foundation), grant number 2020/09850-0; and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq. National Council for Scientific and Technological Development) grant numbers 303210/2017-6 and 431170/2018-5.

Data Availability Statement: the database from the systematic literature review is available upon request from the corresponding author of this paper.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study, in the collection, analyses, or interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

References

1. Nakayama, R.S.; Spínola, M. de M.; Silva, J.R. Towards I4.0: A Comprehensive Analysis of Evolution from I3.0. *Comput. Ind. Eng.* **2020**, *144*, 106453, doi:10.1016/j.cie.2020.106453.
2. Tyrrell, A. Management Approaches for Industry 4.0: A Human Resource Management Perspective. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC); IEEE, 2016; pp. 5309–5316.
3. Lasi, H.; Fettke, P.; Kemper, H.G.; Feld, T.; Hoffmann, M. Industry 4.0. *Bus. Inf. Syst. Eng.* **2014**, *6*, 239–242, doi:10.1007/s12599-014-0334-4.
4. Fragapane, G.; Ivanov, D.; Peron, M.; Sgarbossa, F.; Strandhagen, J.O. Increasing Flexibility and Productivity in Industry 4.0 Production Networks with Autonomous Mobile Robots and Smart Intralogistics. *Ann. Oper. Res.* **2020**, doi:10.1007/s10479-020-03526-7.
5. Yazdi, P.G.; Azizi, A.; Hashemipour, M. An Empirical Investigation of the Relationship between Overall Equipment Efficiency (OEE) and Manufacturing Sustainability in Industry 4.0 with Time Study Approach. *Sustainability* **2018**, *10*, doi:10.3390/su10093031.
6. Mohamed, N.; Al-Jaroodi, J.; Lazarova-Molnar, S. Leveraging the Capabilities of Industry 4.0 for Improving Energy Efficiency in Smart Factories. *IEEE Access* **2019**, *7*, 18008–18020, doi:10.1109/ACCESS.2019.2897045.
7. Brozzi, R.; D'Amico, R.D.; Pasetti Monizza, G.; Marcher, C.; Riedl, M.; Matt, D. Design of Self-Assessment Tools to Measure Industry 4.0 Readiness. A Methodological Approach for Craftsmanship SMEs. *IFIP Adv. Inf. Commun. Technol.* **2018**, *540*, 566–578, doi:10.1007/978-3-030-01614-2_52.
8. Morkovkin, D.E.; Gibadullin, A.A.; Kolosova, E. V.; Semkina, N.S.; Fasehzoda, I.S. Modern Transformation of the

Production Base in the Conditions of Industry 4.0: Problems and Prospects. *J. Phys. Conf. Ser.* **2020**, *1515*, 0–6, doi:10.1088/1742-6596/1515/3/032014.

9. Klingenberg, C.O.; Borges, M.A.V.; Antunes, J.A.V. Industry 4.0 as a Data-Driven Paradigm: A Systematic Literature Review on Technologies. *J. Manuf. Technol. Manag.* **2021**, *32*, 570–592, doi:10.1108/JMTM-09-2018-0325.
10. Adolphs, P.; Epple, U. *Reference Architecture Model Industrie 4.0 (RAMI4.0)*; Berlim, 2015;
11. IVI - Industrial Value Chain Initiative *Industrial Value Chain Reference Architecture (IVRA)*; Chiyoda, 2016;
12. Lin, S.-W.; Murphy, B.; Clauser, E.; Loewen, U.; Neubert, R.; Bachmann, G.; Pai, M.; Hankel, M. Architecture Alignment and Interoperability. *Plattf. Ind. 4.0* **2017**, 19.
13. Schwab, K. *The Fourth Industrial Revolution*; World Economic Forum: Geneva, 2017; ISBN 9781944835019.
14. Castelo-Branco, I.; Cruz-Jesus, F.; Oliveira, T. Assessing Industry 4.0 Readiness in Manufacturing: Evidence for the European Union. *Comput. Ind.* **2019**, *107*, 22–32, doi:10.1016/j.compind.2019.01.007.
15. Lu, Y. Industry 4.0: A Survey on Technologies, Applications and Open Research Issues. *J. Ind. Inf. Integr.* **2017**, *6*, 1–10, doi:10.1016/j.jii.2017.04.005.
16. Hermann, M.; Pentek, T.; Otto, B. Design Principles for Industrie 4.0. *Tech. Univ. Dortmund Fak. Maschinenbau Audi Stift. Supply Net Order Manag.* **2015**, 15.
17. Kagermann, H.; Riemensperger, F.; Hoke, D.; Schuh, G.; Scheer, A.-W. Recommendations for the Strategic Initiative Web-Based Services for Businesses. *Acatech Rep.* **2014**, 112.
18. Xu, L. Da; Xu, E.L.; Li, L. Industry 4.0: State of the Art and Future Trends. *Int. J. Prod. Res.* **2018**, *56*, 2941–2962, doi:10.1080/00207543.2018.1444806.
19. Thomas Bangemann; Christian Bauer; Heinz Bedenbender; Annerose Braune; Christian Diedrich; Markus Diesner; Ulrich Epple; Filiz Elmas; Jens Friedrich; Florian Göbe; Thomas Goldschmidt; Sten Grüner; Martin Hankel; Roland Heidel; Klaus Hesselmann; Guido Hütt; Klein, M.; Löwen, U.; Pfrommer, E.J.; Rauschecker, U.; Schleipen, M.; Schulz, D.; Tasci, T.; Thron, M.; Usländer, T.; et al. *Status Report - Industrie 4.0 Service Architecture - Basic Concepts for Interoperability*; 2016; Vol. 1;.
20. Elmasri, R.; Navathe, S.B. *Sistemas de Banco de Dados*; 4th ed.; Pearson: Londres, 2005; ISBN 9788578110796.
21. Ramakrishnan, R.; Gehrke, J. *Sistemas de Gerenciamento de Banco de Dados*; 3rd ed.; McGraw Hill: Nova Iorque, 2008; Vol. 14; ISBN 9788577803828.
22. Elmasri, R.; Navathe, S.B. *Fundamentals of Database Systems*; 7th ed.; Pearson: Hoboken, 2000; ISBN 9780133970777.
23. Sadalage, P.J.; Fowler, M. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*; Pearson: Hoboken, 2013; ISBN 9780321826626.
24. Codd, E.F. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM* **1983**, *26*, 64–69, doi:10.1145/357980.358007.
25. Özsu, M.T.; Valduriez, P. Distributed and Parallel Database Systems. *Comput. Sci. Handbook, Second Ed.* **2004**, *28*, 58-1-58–24, doi:10.1201/b16768-16.
26. Özsu, M.T.; Valduriez, P. *Principles of Distributed Database Systems*; 3rd ed.; Springer: Londres, 2011; ISBN 9781441988331.
27. Strauch, C.; Kriha, W. Selected Topics on Software-Technology Ultra-Large Scale Site: NOSQL Databases. *Comput. Sci. Media* 2019, 77–84.
28. Moniruzzaman, A.B.M.; Hossain, S.A. NoSQL Database: New Era of Databases for Big Data Analytics - Classification, Characteristics and Comparison. *Int. J. Database Theory Appl.* **2013**, *6*, 1–14, doi:10.1016/S0262-4079(12)63205-9.
29. Abadi, D.J. Consistency Tradeoffs in Modern Distributed Database System Design: CAP Is Only Part of the Story. *Computer (Long. Beach. Calif).* **2012**, *45*, 37–42.
30. Bader, S.; Barnstedt, E.; Bedenbender, H.; Billman, M.; Boss, B.; Braunmandl, A. *Details of the Asset Administration Shell Part 1 - The Exchange of Information between Partners in the Value Chain of Industrie 4.0*; Berlim, 2019;
31. BEDENBENDER, H.; BILLMANN, M.; EPPLE, U.; HADLICH, T.; HANKEL, M.; HEIDEL, H.; HILLERMEIER, O.; HOFFMEISTER, M.; HUHLE, H.; JOCHEM, M.; et al. *Examples of the Asset Administration Shell for Industrie 4.0 Components -*

Basic Part; Frankfurt, 2017;

32. Gastaldi, L.; Appio, F.P.; Corso, M.; Pistorio, A. Managing the Exploration-Exploitation Paradox in Healthcare: Three Complementary Paths to Leverage on the Digital Transformation. *Bus. Process Manag. J.* **2018**, *24*, 1200–1234, doi:10.1108/BPMJ-04-2017-0092.
33. Inigo, M.A.; Porto, A.; Kremer, B.; Perez, A.; Larrinaga, F.; Cuenca, J. Towards an Asset Administration Shell Scenario: A Use Case for Interoperability and Standardization in Industry 4.0. *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. 2020 Manag. Age Softwarization Artif. Intell. NOMS 2020* **2020**, doi:10.1109/NOMS47738.2020.9110410.
34. GAYKO, J. The Reference Architectural Model Rami 4.0 and the Standardization Council as an Element of Success for Industry 4.0 2018.
35. ADOLPHS, P.; AUER, S.; BEDENBENDER, H.; BILLMANN, M.; HANKEL, M.; HEIDEL, R.; HOFFMEISTER, M.; HUHLE, H.; JOCHEM, M.; KIELE-DUNSCHE, M.; et al. *Structure of the Asset Administration Shell: Continuation of the Development of the Reference Model for the Industrie 4.0 Component*; Berlim, 2016;
36. Ye, X.; Hong, S.H. Toward Industry 4.0 Components: Insights into and Implementation of Asset Administration Shells. *IEEE Ind. Electron. Mag.* **2019**, *13*, 13–25, doi:10.1109/MIE.2019.2893397.
37. Bedenbender, H.; Bentkus, A.; Epple, U.; Hadlich, T.; Heidel, R.; Hillermeier, O.; Hoffmeister, M.; Huhle, H.; Kiele-Dunsche, M.; Koziolek, H.; et al. *Industrie 4.0 Plug-and-Produce for Adaptable Factories: Example Use Case Definition, Models, and Implementation*; Berlim, 2017;
38. Gerend, J. Contêineres vs. Máquinas Virtuais Available online: https://docs.microsoft.com/pt-br/virtualization/windowscontainers/about/containers-vs-vm (accessed on 15 June 2021).
39. Xavier, M.G.; Neves, M. V.; Rossi, F.D.; Ferreto, T.C.; Lange, T.; De Rose, C.A.F. Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments. *Proc. 2013 21st Euromicro Int. Conf. Parallel, Distrib. Network-Based Process. PDP 2013* **2013**, 233–240, doi:10.1109/PDP.2013.41.
40. Russmann, M.; Lorenz, M.; Gerbert, P.; Waldner, M.; Justus, J.; Engel, P.; Harnisch, M. Industry 4.0: World Economic Forum. *Bost. Consult. Gr.* **2015**, 1–20.
41. Bechtold, J.; Kern, A.; Lauenstein, C.; Bernhofer, L. *Industry 4.0 - The Capgemini Consulting View*; 2014;
42. Lichtblau, K.; Stich, V.; Bertenrath, R.; Blum, M.; Bleider, M.; Millack, A.; Schmitt, K.; SChmitz, E.; Schröter, M. *Industry 4.0 Readiness*; Aachen, 2015;
43. Bauer, H.; Baur, C.; Camplone, G.; George, K.; Ghislanzoni, G.; Huhn, W.; Kayser, D.; Löffler, M.; Tschiesner, A.; Zielke, A.E.; et al. *Industry 4.0: How to Navigate Digitization of the Manufacturing Sector*; 2015;
44. Petrillo, A.; Felice, F. De; Cioffi, R.; Zomparelli, F. Fourth Industrial Revolution: Current Practices, Challenges, and Opportunities. In *Digital Transformation in Smart Manufacturing*; London, Ed.; Intechopen, 2018; pp. 1–20.
45. Wan, J.; Cai, H.; Zhou, K. Industrie 4.0: Enabling Technologies. *Proc. 2015 Int. Conf. Intell. Comput. Internet Things, ICIT 2015* **2015**, 135–140, doi:10.1109/ICAIOT.2015.7111555.
46. De Mauro, A.; Greco, M.; Grimaldi, M. A Formal Definition of Big Data Based on Its Essential Features. *Libr. Rev.* **2016**, *65*, 122–135, doi:10.1108/LR-06-2015-0061.
47. Russom, P. *Big Data Analytics*; Renton, 2011;
48. Yin, S.; Kaynak, O. Big Data for Modern Industry: Challenges and Trends. *Proc. IEEE* **2015**, *103*, 143–146, doi:10.1109/JPROC.2015.2388958.
49. Kagermann, H.; Wahlster, W.; Helbig, J. Securing the Future of German Manufacturing Industry: Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0. *Final Rep. Ind. 4.0 Work. Gr.* **2013**, 1–84.
50. Gil, D.; Song, I.Y. Modeling and Management of Big Data: Challenges and Opportunities. *Futur. Gener. Comput. Syst.* **2016**, *63*, 96–99, doi:10.1016/j.future.2015.07.019.
51. NIST Big Data Public Working Group *NIST Big Data Interoperability Framework : Volume 6 - Reference Architecture*; Gaithersburg, 2019; Vol. 6;.

52. Tao, F.; Cheng, J.; Qi, Q.; Zhang, M.; Zhang, H.; Sui, F. Digital Twin-Driven Product Design, Manufacturing and Service with Big Data. *Int. J. Adv. Manuf. Technol.* **2018**, *94*, 3563–3576, doi:10.1007/s00170-017-0233-1.
53. Lee, E.A. *Cyber Physical Systems: Design Challenges*; 2017;
54. Gabel, T.; Riedmiller, M. Adaptive Reactive Job-Shop Scheduling With Reinforcement Learning Agents. *Int. J. Inf. Technol. Intell. Comput.* **2008**, *24*, 14–18.
55. Zhang, H.; Stafman, L.; Or, A.; Freedman, M.J. SLAQ: Quality-Driven Scheduling for Distributed Machine Learning. In Proceedings of the Symposium on Cloud Computing; 2017; pp. 390–404.
56. Adam, S.; Buşoniu, L.; Babuška, R. Experience Replay for Real-Time Reinforcement Learning Control. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2012**, *42*, 201–212, doi:10.1109/TSMCC.2011.2106494.
57. Costa, B.S.J.; Angelov, P.P.; Guedes, L.A. Real-Time Fault Detection Using Recursive Density Estimation. *J. Control. Autom. Electr. Syst.* **2014**, *25*, 428–437, doi:10.1007/s40313-014-0128-4.
58. Zhu, H.; Li, M.; Tang, Y.; Sun, Y. A Deep-Reinforcement-Learning-Based Optimization Approach for Real-Time Scheduling in Cloud Manufacturing. *IEEE Access* **2020**, *8*, 9987–9997, doi:10.1109/ACCESS.2020.2964955.
59. Oppitz, M.; Tomsu, P. *Inventing the Cloud Century: How Cloudiness Keeps Changing Our Life, Economy and Technology*; Springer: Cham, 2018; ISBN 9783319611600.
60. Rubin, V.L. Veracity Roadmap: Is Big Data Objective, Truthful and Credible? *Adv. Classif. Res. Online* **2014**, *24*, 4–15, doi:10.7152/acro.v24i1.14671.
61. Mavridis, I.; Karatza, H. Combining Containers and Virtual Machines to Enhance Isolation and Extend Functionality on Cloud Computing. *Futur. Gener. Comput. Syst.* **2019**, *94*, 674–696, doi:10.1016/j.future.2018.12.035.
62. Oliveira, V.F. De; Pinheiro, E.; Daniel, J.F.L.; Guerra, E.M.; Junqueira, F.; Santos Fo, D.J.; Miyagi, P.E. Infraestrutura de Dados Para Sistemas de Manufatura Inteligente. In Proceedings of the 14th IEEE International Conference on Industry Applications; IEEE: São Paulo, 2021; pp. 516–523.
63. Fowler, M. Reporting Database Available online: https://martinfowler.com/bliki/ReportingDatabase.html (accessed on 15 June 2021).
64. Han, J.; Haihong, E.; Le, G.; Du, J. Survey on NoSQL Database. *Proc. - 2011 6th Int. Conf. Pervasive Comput. Appl. ICPCA 2011* **2011**, 363–366, doi:10.1109/ICPCA.2011.6106531.
65. Cattell, R. Scalable SQL and NoSQL Data Stores. *SIGMOD Rec.* **2010**, *39*, 12–27, doi:10.1145/1978915.1978919.
66. Khine, P.P.; Wang, Z. A Review of Polyglot Persistence in the Big Data World. *Information* **2019**, *10*, doi:10.3390/info10040141.
67. Cavalieri, S.; Salafia, M.G. A Model for Predictive Maintenance Based on Asset Administration Shell. *Sensors (Switzerland)* **2020**, *20*.
68. Lang, D.; Grunau, S.; Wisniewski, L.; Jasperneite, J. Utilization of the Asset Administration Shell to Support Humans during the Maintenance Process. *IEEE Int. Conf. Ind. Informatics* **2019**, *2019-July*, 768–773, doi:10.1109/INDIN41052.2019.8972236.
69. Ye, X.; Jiang, J.; Lee, C.; Kim, N.; Yu, M.; Hong, S.H. Toward the Plug-and- Produce Capability for Industry 4.0. *IEEE Ind. Electron. Mag.* **2020**, *14*.
70. Foundation, O. OPC 10000-6: OPC Unified Architecture - Part 6: Mappings 2017.
71. Erl, T.; Khattak, W.; Buhler, P. *Big Data Fundamentals: Concepts, Drivers & Techniques*; Prentice Hall: Hoboken, 2016; ISBN 0975442201.
72. Stonebraker, M. SQL Databases v. NoSQL Databases. *Commun. ACM* **2010**, *53*, 10–11, doi:10.1145/1721654.1721659.
73. Inmon, W.H. *Building the Data Warehouse*; 3rd ed.; John Wiley & Sons: Hoboken, 2002; ISBN 0-471-08130-2.
74. Thomsen, E. *OLAP Solutions: Building Multidimensional Information Systems*; 2nd ed.; John Wiley & Sons, 2002; ISBN 0471400300.
75. Bicevska, Z.; Oditis, I. Towards NoSQL-Based Data Warehouse Solutions. *Procedia Comput. Sci.* **2016**, *104*, 104–111,

doi:10.1016/j.procs.2017.01.080.

76. Chevalier, M.; Malki, M. El; Kopliku, A.; Teste, O.; Tournier, R. Implementing Multidimensional Dasta Warehouse into NoSQL. In Proceedings of the 17th International Conference on Enterprise Information Systems; Barcelona, 2015.
77. Yangui, R.; Nabli, A.; Gargouri, F. Automatic Transformation of Data Warehouse Schema to NoSQL Data Base: Comparative Study. *Procedia Comput. Sci.* **2016**, *96*, 255–264, doi:10.1016/j.procs.2016.08.138.
78. Fatima, H.; Wasnik, K. Comparison of SQL, NoSQL and NewSQL Databases for Internet of Things. *IEEE Bombay Sect. Symp. 2016 Front. Technol. Fuelling Prosper. Planet People, IBSS 2016* **2016**, doi:10.1109/IBSS.2016.7940198.
79. Ali, W.; Shafique, M.U.; Majeed, M.A.; Raza, A. Comparison between SQL and NoSQL Databases and Their Relationship with Big Data Analytics. *Asian J. Res. Comput. Sci.* **2019**, *4*, 1–10, doi:10.9734/ajrcos/2019/v4i230108.