Graph Databases for Multi-Domain Taxonomies in Maritime Systems

Janica A. Bronson[1], Fernando H. P. Luz[1], Ícaro A. Fonseca[1], and Henrique M. Gaspar[1]

[1]Department of Ocean Operations and Civil Engineering, Norwegian University of Science and Technology, e-mail: henrique.gaspar@ntnu.no

KEYWORDS

Graph Database, Multi-domain Taxonomies, Maritime Systems

ABSTRACT

Facing the increasing quantity and richness of data related to modeling and simulations of maritime systems, advanced solutions that can maintain data coherence and traceability are becoming increasingly beneficial. In the context of modeling complex systems, such as ship design, retaining connections of systems data, whether quantitative or qualitative, helps to provide a traceable and comprehensive view of the vessel at various stages of its development. This paper explores the use of graph databases to help provide such connections and introduces various domains of systems data that would be advantageous to link via graph theory. Additionally, this paper explores how graph databases compare to current flat file databases in a practical sample case of multi-domain data related to ship concept design.

Multi-Domain Taxonomies in Maritime Systems

Arranging information according to certain rules is a common way to interpret and synthesize information. As noted by Simon (1962), this is a key element that enables us to solve complex problems. Especially in engineering analysis and problem-solving, reorganizing a taxonomy into other schemes may be useful for better contextualization. As the raw information is the same, elements from one taxonomy to another can be easily translated via a dictionary. This parsing from abstract and qualitative towards quantification is a core fundamental aspect in engineering and, consequently, computer-aided methods.

Within taxonomies, translations of system elements by common attributions or domains can lead to an infinite regress of sub-categorizations. This presents a challenge on how we can continue to facilitate the regrouping, parsing, and reinterpreting of systems information without losing semantic binds from information retranslation.

Maritime systems will be used for the rest of this paper as a case study to explore how to manage multi-domain taxonomies. The authors believe, however, that the discussion here can be useful to large extensions of other systems, especially for physically large and complex operations.

In this paper, taxonomies are understood as a hierarchical classification of system elements (Simon, 1962), and domains are considered logical subcategorizations within taxonomies covering either tangible components, phenomena or behaviors, and contextual information. Using domains can be a useful way to categorize and manage information within complex systems. These data domains can also be understood as various perspectives and presentations of an object, analogous to 'aspects' as defined in IEC 81326 for Industrial Systems (ISO, 2022) that cover events, processes, products, or the functions of a system. Central to the idea of domains includes the fact that information within a domain can be linked to other domains and can be reinterpreted based on the users' needs.

In the shipping industry, a vessel's data can be translated into various domains, from design to operations. At the design level, a ship's systems are functionally interpreted, but during construction, the focus shifts to a product orientation. Task breakdowns that combine process and location domains are heavily used during the ship construction phase to coordinate materials and labor in the yard. Reorientation back to the functional domain occurs upon the vessel's delivery to permit overall ship evaluation and systems testing (Pal, 2015).

We converge to six discrete data domains, defined in detail in the next section, that are useful to note within a ship's taxonomy.

Maritime Design and Construction Domains

Functional Domain: Functional data is information related to parameters or attributes that describe the system's intended purpose (Pettersen, 2018). Functional ship breakdowns are some of the most popular ways to decompose a vessel today, where the two most widely-recognized structures include the Ship Work Breakdown Structure (SWBS) and the Skipsteknisk Forskningsinstitutt (SFI) Group system. These systems decompose the vessel by the hull structure, propulsion plant, electric plant, auxiliary systems, outfitting, and furnishing, among others Koenig et al. (1997). Data involved in designing, defining, and evaluating ship systems fall within this category. For example, to describe propulsion systems in a ship, supplemental data can range from vessel performance parameters all the way to resistance test results.

Product Domain: Product data refers to information about the physical form of ship parts that perform specific functions (Pettersen, 2018). This type of representation is commonly used in the detailed engineering and construction phases for ships. Tangible assets in this domain include the combination of the ship hull and the ship outfitting, which comprises various equipment and furnishings. During ship construction, these components are typically separated based on their physical connections. These zones or blocks are made up of steel blocks, systems, and their respective parts or components (Pal, 2015).

Schedule or Process Domain: Process data refers to information related to workflows involved in ship product data production, encompassing both administrative tasks such as planning and scheduling, as well as technical tasks. Multiple administrative and technical processes govern the design and construction of a ship due to its high complexity, and tasks are typically divided into engineering sub-teams that perform their work concurrently (NAVSEA, 2012). Each team will perform planning, checking, and reviewing activities. Similarly, shipbuilding processes require complex planning and documentation to streamline the coordination of labor and materials.

Human Domain: People data refers to information related to individuals involved in executing tasks and ship design and building activities. Due to the high degree of multi-organization in the shipping industry (Emblemsvåg, 2014), a significant amount of data can be collected concerning all the parties involved in ship design and operations. The requirements for gathering personnel data may vary throughout the lifecycle of a vessel, from ensuring that individuals involved in its design possess valid security compliance to ensuring that the persons operating the ship meet health and certification requirements.

Geospatial Domain: Geospatial data refers to information that is related to the physical location of tangible ship assets as well as material and labor resources. Since there are numerous stakeholders involved in a ship's lifecycle, and they are spread across the globe, personnel and ship assets are also highly dispersed. In cases where shipbuilding offshoring is prevalent, such as in the case of European shipbuilders, this dispersion is even more pronounced (Semini et al., 2023). During ship operations, geospatial data also plays a critical role as the ship's Automatic Identification Systems (AIS) track and monitor the position of a vessel for global fleet traffic management.

Contextual Domain: Contextual data refers to information that helps in making decisions. It involves background data that provides a high-level understanding of other data domains. Contextual awareness is crucial in situations where uncertainty is high. In the shipping industry, uncertainties exist in multiple dimensions, ranging from business markets to vessel behavior. Knowledge management systems that facilitate context-aware decision-making in the industry would be useful in supplementing tacit enterprise and technical knowledge. Currently, communication tools like Power BI are increasingly popular for data-driven decision-making.

Handling Multi-Domain Taxonomies

In current practice, information within any of these domains is generated and processed in an extremely heterogeneous manner. Coordinating the technical design as well as managing the resources, materials, and time to ensure the completion and quality of a ship concept is a massive undertaking from a data management perspective.

Solutions to manage and store data in these multiple domains may range from flat file databases developed in-house via MS Excel to advanced Enterprise Resource Planning (ERP) software that uses relational databases. However, where databases and storage are concerned, the usage of these solutions is varied given their limitations in realizing an integrated view with process, people, and contextual domains outside technical product or functional data.

Before proposing a way to address the multi-domain taxonomies presented in the previous section, it is important to remember some issues and critical considerations for the proposal:

Data Replication: In our solution, it is critical to avoid data replication. Every time a value needs to be updated, it is necessary to iterate in the entire system and apply the new value, and the data will be inconsistent until the update is completed. Another side effect is the waste of storage space using this approach.

Domain Element Identification: Each domain uses a specific denomination of an element. In a proposal to handle these taxonomies, it is important to not affect this characteristic in a domain.

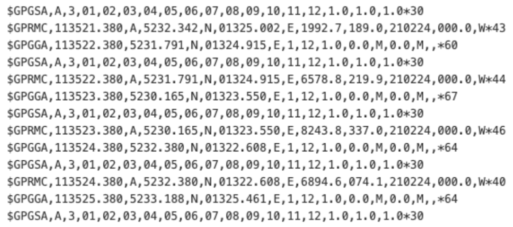
Less Intrusive: A good approach can not change the current tools and system in an intrusive way. Creating a middleware with a proper API (Application Programming Interface) dealing with each system is important.

With this in mind, the following section discusses possible data models that can be used to handle the multi-domain ship taxonomy.

Flat File Database

A flat file database is a collection of data in two dimensions (rows and columns) stored in a plain text format, where each row is a separate record. Generally, flat files are a good choice in two scenarios. The first scenario is in early-stage projects due to the simplicity in maintenance these databases offer. However, flat file databases can start to be prohibitive with increasing complexity and quantity of data. The second scenario is when there is a need to append large amounts of data, and the flat file database provides a low overhead to inserting new information in raw format. Log system files are a good example of this scenario.

One example of a flat file database in the maritime field is the logs and messages of NMEA-0183 data format (Betke, 2001). In NMEA-0183 (v 3.01), all messages begin with **$** and end with a carriage return `\c` and a line feed `\n`. Data fields follow comma **,** delimiters and are variable in length. Null fields still follow comma **,** delimiters but contain no information. An asterisk $*$ delimiter and checksum value follow the last field of data contained in an NMEA-0183 message. The checksum is the 8-bit exclusive of all characters in the message, including the commas between fields, but not including the **$** and asterisk delimiters. The hexadecimal result is converted to two ASCII characters (0–9, A–F), where the most significant character appears first.



```
$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30
$GPRMC,113521.380,A,5232.342,N,01325.002,E,1992.7,189.0,210224,000.0,W*43
$GPGGA,113522.380,5231.791,N,01324.915,E,1,12,1.0,0.0,M,0.0,M,,*60
$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30
$GPRMC,113522.380,A,5231.791,N,01324.915,E,6578.8,219.9,210224,000.0,W*44
$GPGGA,113523.380,5230.165,N,01323.550,E,1,12,1.0,0.0,M,0.0,M,,*67
$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30
$GPRMC,113523.380,A,5230.165,N,01323.550,E,8243.8,337.0,210224,000.0,W*46
$GPGGA,113524.380,5232.380,N,01322.608,E,1,12,1.0,0.0,M,0.0,M,,*64
$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30
$GPRMC,113524.380,A,5232.380,N,01322.608,E,6894.6,074.1,210224,000.0,W*40
$GPGGA,113525.380,5233.188,N,01325.461,E,1,12,1.0,0.0,M,0.0,M,,*64
$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30
```

Figure 1: Example of a data structure NMEA message.

Relational Database Model

A relational database is composed of tables, rows, and columns, where it is possible to create relationships between data by joining tables. In a relational database, each row in the table is a record with a unique ID called *primary key*. The columns of the table hold attributes of the data, and each record usually has a value for each attribute, enabling a way to create relationships among data points.

Benefits

Relational nature: Relational databases allow easy querying on relationships between data among multiple tables, and these table relationships are important for effectively organizing, composing, and structuring diverse data.

ACID compliant: In general, relational databases support ACID (Atomicity, Consistency, Isolation, Durability), ensuring data validity. That means:
– **Atomic**: One is able to execute *all* or *nothing* transactions;
– **Consistent**: The data to insert must be valid, and this is assured by the execution of rules;
– **Isolated**: Multiple transactions can be executed at the same time without interference between them;
– **Durable**: Once the transaction is concluded, it is possible to retrieve committed data.

Data is structured: The data is well-structured and minimizes the occurrences of potential errors due to the fact the Structured Query Language (SQL) schema requires that the data model and the format of the data is known before storing it.

Disadvantages

Structure must be created in advance: Structured data can lead to fewer errors, but it also means that columns and tables have to be created ahead of time. Hence, relational databases take more time to set up, and it is important to be aware of the entire system beforehand. They are not effective for storing and querying unstructured data, where the format is unknown and may change over time.

Difficult to scale: Relational databases are difficult to scale horizontally (spread in several instances) because of their relational nature. For read-heavy systems, it is straightforward to provision multiple read-only replicas. However, the only option for write-heavy systems is to scale vertically by improving instance specifications, such as memory, storage, and processor. This is generally more expensive than provisioning additional servers.

Non-Relational Database Model

A non-relational database (also known as No-SQL) is a model that stores data in an unstructured form. The way to store them is optimized for the specific requirements of the data type. In this model, data can be stored in key/value pairs, JSON documents, or as a graph consisting of edges and vertices. The advantages and disadvantages of these models are discussed below.

Benefits

More flexibility: Non-relational databases are more flexible and simpler to set up as they do not support table relationships, and data is usually stored in documents or as key/value pairs. Due to this, they are a better choice for storing unstructured data or when all relationships in the system are not known upfront.

Data sharding: Sharding is a procedure to increase data space separately across multiple computers, also known as horizontal scaling. This enables an increased capacity to deal with large amounts of data without the necessity to purchase powerful but expensive servers.

Peer-to-peer replication: Non-relational databases are typically designed for distributed use cases and heavy-write systems that allow peer-to-peer replication or the ability to have multiple writes in the same data partition.

Disadvantages

Eventual consistency: A trade-off of peer-to-peer replication is a loss of strong consistency after writing to a shard in a distributed non-relational database cluster. In this case, a small delay is possible before that write update can be propagated to other replicas, and, during this time, reading from a replica can mean accessing stale data. This situation is called eventual consistency. This affects distributed databases in general, not only non-relational databases. A single shard in a non-relational database can be strongly consistent. Still, to fully take advantage of the scalability benefits, the non-relational database must be set up as a distributed cluster.

Lack of ACID: Non-relational databases do not support ACID (atomicity, consistency, isolation, durability) transactions across multiple documents. It is possible to achieve single-record atomicity with an appropriate schema design, which is acceptable for a lot of applications. However, there are still many applications that require ACID across multiple records.

Data replication: Due to the absence of transactions across tables, data models in non-relational databases generally encounter data duplication. However, storage is cheap, and most people consider this to be a minor drawback.

Graph Database Model

Graph Database Model is a subdomain of a non-relational database, where the focus of this database is on the relationships between the data. The database queries and infrastructure are adapted to retrieve the data using these relationships in a faster way. The data organization for this database uses the graph theory concepts.

Graph Theory Overview and Types

Graph theory is the study of network properties based on the connections between nodes. A graph consists of a set of *vertices* (nodes or points) and a set of *edges* (lines or connections) representing the links between these vertices as shown in Figure 2. Mathematically, it is a non-linear structure represented by $G = (V, E)$, where V is the set of vertices and E is the set of edges. The number of vertices in a graph is called its *order*, denoted as $|V|$, and the number of edges is its *size*, denoted as $|E|$.

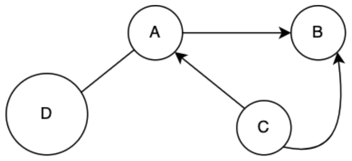


Figure 2: Illustration of a graph

Graphs are classified based on their characteristics, such as *Directed Graphs*: Graphs with directed edges, i.e., with a direction between vertices; *Weighted Graphs*: Graphs where each edge has an associated weight; *Connected Graphs*: Graphs where there is a path between every pair of vertices. *Acyclic Graphs*: Graphs with no cycles (a sequence of edges that begins and ends at the same vertex).

Graphs can be represented using various methods, including adjacency matrices and adjacency lists. These representations are crucial for implementing graph algorithms and efficiently storing graph-related data. When properly represented, graph theory serves as a powerful mathematical framework for modeling and analyzing relationships in various real-world scenarios. Its rich theoretical foundation and practical applications make graph theory an essential area of study in both mathematics and computer science.

Graph Theory Applications

Today, graph theory is used to address various real-world problems, like optimizing routes and logistics for roads and airline networks, analyzing social interaction and influence behaviors in social networks, and detecting malicious activity patterns in cybersecurity on the internet. Due to the dynamic structure of graphs and efficient data querying, graph databases have a broadened appeal across various business and industry sectors. A few notable applications that have helped to popularize graph theory are listed below Robinson et al. (2015).

Networks: Modeling relationships in social networks, communication networks, and transportation systems.

Computer Science: Representing data structures (e.g., adjacency matrices and lists) and solving route planning and optimization.

Biology: Analyzing biological networks, such as protein-protein interaction.

Operations Research: Solving problems related to resource allocation, scheduling, and optimization.

In shipbuilding and other complex engineer-to-order (ETO) industries, graph theory can help address the problem related to multi-domain taxonomies by providing a mechanism to map dependencies between processes, temporal data, and functional ship breakdowns. By enabling the linkages across domains, it is possible to represent a single data node from different domains, helping to ensure the uniqueness of domain element identification. Figure 3 illustrates this idea where domains are visualized as different layers. The benefits of applying graph theory are described below.

- **Chain effect of changing parameters across multi-domains:** Different domains may have some dependency between them, and a parameter change in one domain may significantly affect another domain. Graph theory can provide an explicit dependency between domains and trigger the other domains when a parameter change occurs.
- **Multi-domain consistency:** Data coherence across all domains in the shipbuilding process can help to enable accurate data retrieval not only for technical work but also for general business operations.

Graph Database on Multi-Domain Taxonomies

To evaluate the potential of graph databases compared to flat file and relational databases, a sample case is developed in this section using the domains mentioned, except for the contextual domain. In this

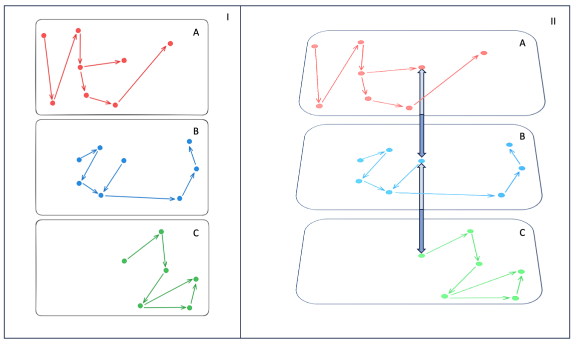


Figure 3: I) Three example domains are represented by a graph, where A and B have 8 connections, and C has 6 connections. II) The same domains are presented, where the arrows indicate the dependency for a specific node that can be connected across all layers.

example, only the scope of the concept design activities defined by Andrews (1985) was considered to keep the data concise and legible for evaluation.

In the example, the data represents the design activities involved in completing a single iteration of a ship concept. The data contains 35 unique records of activities associated with ship functions such as Vessel Performance, Vessel Stability, and Mission Systems. These activities are also assigned to randomized personnel (for this study) for completion within a certain deadline and have unique attributes related to the type of data (digital or non-digital) and file (document, drawing, or model) to be delivered. Hence in this example, the following main objects are used according to their related domains:

- **Schedule**: MonthDue (deadline of activity), ShipLC (associated lifecycle phase)
- **Functional**: Systems (systems associated with activity)
- **Product**: Description (definition of activity)
- **Persons**: NameResponsible (name of assigned personnel), Team (team of assigned personnel)
- **Location**: BasedAt (location of assigned personnel)

where the relationships between these objects are:

- PARTOF – Relationship of activity (Description) to ship systems (Systems)
- WITHINLCPHASE – Ship systems (Systems) relative to ship's lifecycle phases (ShipLC)
- DUEBYMONTH – Activity (Description) due by a certain deadline (MonthDue)
- ASSIGNEDTO – Assignee (NameResponsible) responsible for the activity (Description)
- MEMBEROF – Assignee (NameResponsible) as part of a team (Team)
- BASEDAT – Location (BasedAt) of the assignee (NameResponsible)

As the intent of this example is not completion, the contents are based on an understanding of ship departments (NAVSEA, 2012) but are not representative of any particular company or project.

Flat File Representation

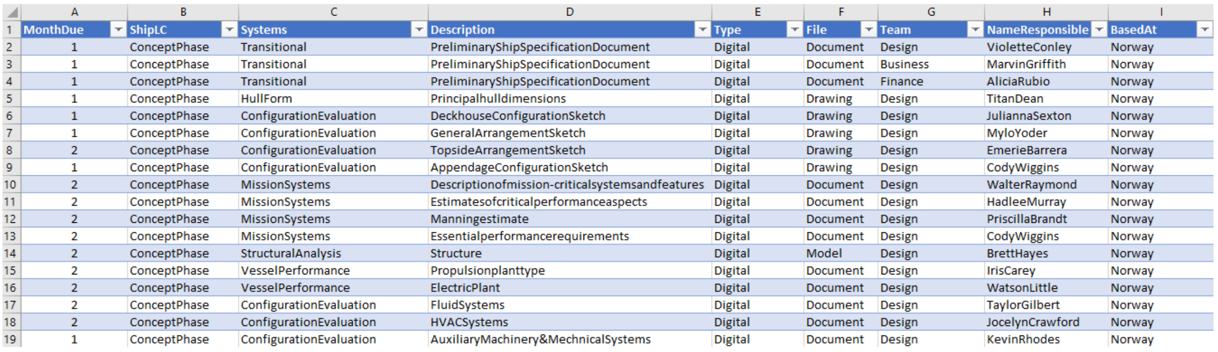
Figure 4 provides a flat file database representation of this case, which is a table generated in MS Excel. Data in this example is stored as unique observations per row, with columns as the main objects. Although this example was simple to generate and manipulate, relationships are not explicitly represented, and users must infer such connections. Along with the inability to recognize such relationships, data redundancy and inconsistencies are not easy to detect, so information can be easily compromised. Performing complex extraction, transformation, and loading (ETL) functions can be challenging to perform securely, as they run the risk of corrupting the data.

Graph Database Representation

To present a graph database of the sample case and deploy a visual representation, ArangoDB was used. It is a scalable graph database system that houses an Insights Platform for managing and running graph database information and visualizations (ArangoDB, 2024b). ArangoDB also flexibly supports JSON formats, which were used as input or data sources for the platform. In one JSON file, we defined the relationships used to describe the edges, while a second JSON file was used to populate the nodes with the object keys. These files were generated from a Python script that converted data analogous to the MS Excel example described in the previous section. A JavaScript code was then used to process these JSON inputs and develop node and edge collections. Relations are also explicitly defined for the purposes of visualization.

Figure 5 provides a partial view of the generated visual. In this case, each unique observation is part of a node collection that can be visualized and color-coded in ArangoDB's graph view. In this example, the colors help to distinguish between the various domains, where the respective domains represented are magenta for the people domain, orange for the geospatial domain, red for the product domain, blue for the functions or systems domain, and green for the temporal and scheduling domain.

A total of 130 edges and 68 nodes were recognized. In this example, it is possible to clearly see the connections of nodes across domains, and planning information (related to scheduling and person domains) can be tangibly viewed in relation to technical details (such as ship functions). With the benefits of parameter change chain effects, this view can be helpful in engineering change management where any updates to design activity nodes (or the Description objects in this scenario) will always be seen in connection to



	A	B	C	D	E	F	G	H	I
1	MonthDue	ShipLC	Systems	Description	Type	File	Team	NameResponsible	BasedAt
2	1	ConceptPhase	Transitional	PreliminaryShipSpecificationDocument	Digital	Document	Design	VioletteConley	Norway
3	1	ConceptPhase	Transitional	PreliminaryShipSpecificationDocument	Digital	Document	Business	MarvinGriffith	Norway
4	1	ConceptPhase	Transitional	PreliminaryShipSpecificationDocument	Digital	Document	Finance	AliciaRubio	Norway
5	1	ConceptPhase	HullForm	Principalhulldimensions	Digital	Drawing	Design	TitanDean	Norway
6	1	ConceptPhase	ConfigurationEvaluation	DeckhouseConfigurationSketch	Digital	Drawing	Design	JuliannaSexton	Norway
7	1	ConceptPhase	ConfigurationEvaluation	GeneralArrangementSketch	Digital	Drawing	Design	MyloYoder	Norway
8	2	ConceptPhase	ConfigurationEvaluation	TopsideArrangementSketch	Digital	Drawing	Design	EmerieBarrera	Norway
9	1	ConceptPhase	ConfigurationEvaluation	AppendageConfigurationSketch	Digital	Drawing	Design	CodyWiggins	Norway
10	2	ConceptPhase	MissionSystems	Descriptionofmission-criticalsystemsandfeatures	Digital	Document	Design	WalterRaymond	Norway
11	2	ConceptPhase	MissionSystems	Estimatesofcriticalperformanceaspects	Digital	Document	Design	HadleeMurray	Norway
12	2	ConceptPhase	MissionSystems	Manningestimate	Digital	Document	Design	PriscillaBrandt	Norway
13	2	ConceptPhase	MissionSystems	Essentialperformancerequirements	Digital	Document	Design	CodyWiggins	Norway
14	2	ConceptPhase	StructuralAnalysis	Structure	Digital	Model	Design	BrettHayes	Norway
15	2	ConceptPhase	VesselPerformance	Propulsionplanttype	Digital	Document	Design	IrisCarey	Norway
16	2	ConceptPhase	VesselPerformance	ElectricPlant	Digital	Document	Design	WatsonLittle	Norway
17	2	ConceptPhase	ConfigurationEvaluation	FluidSystems	Digital	Document	Design	TaylorGilbert	Norway
18	2	ConceptPhase	ConfigurationEvaluation	HVACSystems	Digital	Document	Design	JocelynCrawford	Norway
19	1	ConceptPhase	ConfigurationEvaluation	AuxiliaryMachinery&MechnicalSystems	Digital	Document	Design	KevinRhodes	Norway

Figure 4: Flat File Database with Sample Case Data represented in MS Excel

relevant team members and understood to affect related ship functions. Aside from engineering change management, the flexibility of incorporating details by adding more and more keys to a node makes this solution adaptable to changing levels of detail (LOD). This can be helpful as a means to view high-level ship catalog data and references, as well as granular ship design details in a specific project while maintaining coherence and consistency.

Discussion

While both databases show how multi-domain data are handled, we can compare them in terms of the considerations described in the earlier sections: data replication, domain element identification, and level of intrusion. Other notable differences to consider are discussed in the following sections.

In terms of data replication and intrusion, the flat file database requires direct interaction to update any information. Creating a copy or using read-only versions are the only means to provide a level of security; otherwise, it is easy to corrupt the data. Meanwhile, for the graph database implementation, data manipulations such as filtering are done within the ArangoDB platform, which is separate from the data source. Due to the unstructured nature of graph databases, data replication can happen, unfortunately, but can be managed by scripts for data transformations such as the one developed in this case study.

Scalability

As this example expands outside the rather small scope explored in this case scenario, both graph databases and flat file databases may face scalability issues. For graph databases, an increasing volume of nodes may also lead to more dynamic relationships (Pokorný, 2015) given the richness of data from unique labels and properties attached to various nodes and edges (Besta et al., 2023). ArangoDB has a limit of up to 4000 execution nodes (ArangoDB, 2024a) and other complexity limitations that may pose constraints to ship design projects that have data of sizes of up to ten gigabytes (Whitfield et al., 2011).

As previously discussed, non-relational databases such as graph databases can typically be scaled horizontally through sharding, while relational databases can typically be scaled vertically (Wang and Yang, 2017). Having a database spread across multiple servers means that there is a high amount of network requests to be expected (ArangoDB), as well as a reduction in query execution performance. The latter is a challenge, especially when one expects users to be accessing data from the database in multiple instances. Partitioning graph data and handling multiple queries or transactions from a single source leads to higher processing time. These types of transactions are called built-in parallelism, and platforms, including ArangoDB, are attempting to provide more efficient solutions to tackle this challenge.

While both means of scaling will require more resources and costs, horizontal scaling is often less expensive and reduces the risk of a 'single point of failure' but may require more resources for maintenance. In the case of graph databases, the capability to scale vertically will strongly vary based on the graph database platform being considered. ArangoDB supports both vertical and horizontal scaling but encourages the use of horizontal scaling to increase resilience, performance, and fault tolerance (ArangoDB).

Open Collaborative Platforms and Interface

To interface with these databases, more user-friendly and maintainable custom scripts will also need to be developed. These scripts can help to automate functions such as entry, data sharding, and retrieval or querying of data from these databases. Without these tools or proper data governance rules and infrastructure, the disadvantages that come with non-relational databases may also increase technical debt to fix data duplication or data errors. Naturally, these challenges increase with scale.

Apart from queries, challenges related to distribution, specifically how information can be shared across multiple servers, are critical considerations ship design firms and yards need to account for. According to Besta et al. (2023), there is currently no data distribution scheme for graph databases in practice and limited knowledge of the performance of a highly distributed graph analytics framework. Hence, a graph database infrastructure that can accommodate diverse and geographically dispersed stakeholders, as in the case of

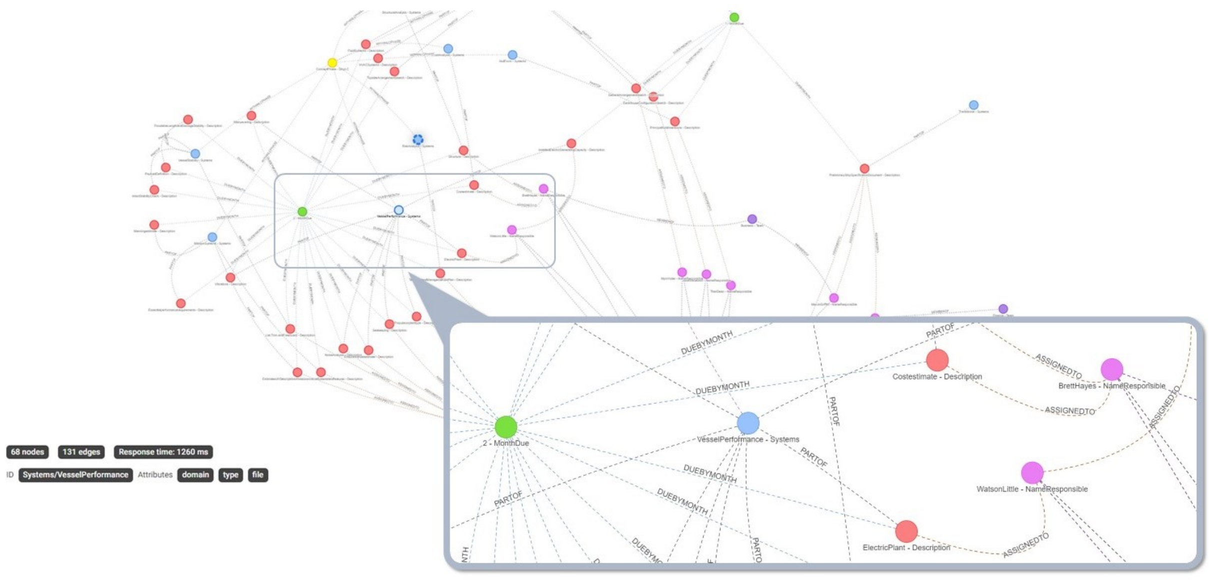

68 nodes
131 edges
Response time: 1260 ms
ID Systems/VesselPerformance Attributes domain type file
DUEBYMONTH
PARTOF
Costestimate - Description
ASSIGNEDTO
BrettHayes - NameResponsible
2 - MonthDue
VesselPerformance - Systems
WatsonLittle - NameResponsible
ElectricPlant - Description

Figure 5: Graph Database Representation for Sample Case

shipbuilding, will require specialized infrastructure considerations related to data centers and routing architecture.

As managing and interacting with such databases' distribution infrastructure becomes challenging, considering how open and collaborative platforms to address such gaps may be paramount. For example, one compelling feature of graph databases is their ability to potentially manage unstructured and semi-structured data (Soussi et al., 2011), which most flat file and defined relational databases are unable to tackle. Graphs provide a natural way of organizing data in a unified and contained manner. In an enterprise context, where data may come in emails, audio, and other diverse modalities, graphs can serve as a mechanism for mapping data from all these sources. This can ultimately be a stepping-stone for realizing a ship's multi-domain taxonomy and enabling more collaborative knowledge management throughout a ship's lifecycle.

Future Work

In order to assess the scalability of this solution, a more detailed and technical database shall be developed incorporating vessel information beyond concept design. Solutions and other emerging technologies to tackle scalability issues shall be evaluated, along with the possible development of a platform for the migration and management of data from multiple sources into an integrated Single Source of Truth (SSoT) graph database.

Concluding Remarks

In this paper, the potential of graph databases to manage multi-domain taxonomy in shipbuilding was explored. These benefits can facilitate improved data management and retrieval across multiple domains, which may ultimately help to prevent data errors during engineering planning and execution. Outside shipbuilding, and especially in the context of modeling and simulations for complex systems, these benefits are considered transferable as information related to a system is not only limited to physical dimensions. Incorporating multiple data domains can provide a more complete and comprehensive digital understanding of such systems.

ACKNOWLEDGEMENTS

This work was done in close collaboration with the EU HORIZON SEUS Project, where a Docker instance of ArangoDB is jointly used with consortium partners such as SARC BV.

References

David Andrews. An Integrated Approach to Ship Synthesis. *Royal Institution of Naval Architects*, 1985. ISSN 0306-0209.

ArangoDB. The Scalability of ArangoDB and Its Data Models (3.11 Release Documents). URL `https://docs.arangodb.com/3.11/deploy/architecture/scalability/`.

ArangoDB. Known Limitations for AQL Queries, 2024a. URL `https://docs.arangodb.com/3.12/aql/fundamentals/limitations/`.

ArangoDB. What is ArangoDB?, 2024b. URL `https://docs.arangodb.com/3.11/about-arangodb/`.

MacIej Besta, Robert Gerstenberger, Emanuel Peter, Marc Fischer, Michał Podstawski, Claude

Barthels, Gustavo Alonso, and Torsten Hoefler. Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries. *ACM Computing Surveys*, 56(2), 9 2023. ISSN 15577341. doi: 10.1145/3604932.

Klaus Betke. The NMEA 0183 protocol. *Standard for Interfacing Marine Electronics Devices*, 2001.

Jan Emblemsvåg. Lean project planning in shipbuilding. *Journal of Ship Production and Design*, 30(2):79–88, 2014. ISSN 21582874. doi: 10.5957/JSPD.30.2.130054.

ISO. IEC 81326-1 Industrial systems, installations and equipment and industrial products - Structuring principles and reference designations - Part 1: Basic rules. Technical report, ISO, 2022. URL `https://www.iso.org/standard/82229.html`.

Philip Koenig, Peter MacDonald, Thomas Lamb, and John Dougherty. Towards a Generic Product-Oriented Work Breakdown Structure For Shipbuilding. Technical report, Louisiana, 4 1997.

NAVSEA. Ship Design Manager (SDM) and Systems Integration Manager (SIM) Manual. Technical report, 2012.

Malay Pal. Ship Work Breakdown Structures through Different Ship Lifecycle Stages. In *International Conference on Computer Applications in Shipbuilding*, 2015.

Sigurd Solheim Pettersen. Resilience by Latent Capabilities in Marine Systems, 2018.

Jaroslav Pokorný. Graph databases: Their power and limitations. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9339:58–69, 2015. ISSN 16113349. doi: 10.1007/978-3-319-24369-6{_}5.

Ian Robinson, Jim Webber, and Emil Eifrem. *Graph Databases: New Opportunities for Connected Data*. O'Reilly Media, Inc., 2nd edition, 2015. ISBN 1491930896.

Marco Semini, Clara Patek, Per Olaf Brett, Jose Jorge Garcia Agis, Jan Ola Strandhagen, and Jørn Vatn. Some relationships between build strategy and shipbuilding time in european shipbuilding. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 237(3):658–676, 2023. doi: 10.1177/14750902221141749. URL `https://doi.org/10.1177/14750902221141749`.

Herbert A. Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6):467–482, 1962. ISSN 0003049X. URL `http://www.jstor.org/stable/985254`.

Rania Soussi, Marie-Aude Aufaure, and Hajer Baazaoui. *Graph Databases for Collaborative Communities*. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-19046-9. doi: 10.1007/978-3-642-19047-6. URL `https://link.springer.com/10.1007/978-3-642-19047-6`.

Ruihan Wang and Zong-Mou Yang. Sql vs nosql: A performance comparison. 2017. URL `https://api.semanticscholar.org/CorpusID:209375121`.

R. I. Whitfield, A. H.B. Duffy, P. York, D. Vassalos, and P. Kaklis. Managing the exchange of engineering product data to support through life ship design. *CAD Computer Aided Design*, 43 (5):516–532, 5 2011. ISSN 00104485. doi: 10.1016/j.cad.2010.12.002.

AUTHOR BIOGRAPHIES

JANICA A. BRONSON is a current PhD Candidate for Maritime Computational Tools at NTNU. She completed a Master's in Naval Architecture at the University of British Columbia and a Bachelor of Mechanical Engineering at the University of Calgary. Before NTNU, she worked in industry at Robert Allan and Vard Marine, focusing on ship concept design and programmatic tools for data handling.

FERNANDO H. P. LUZ is an Engineering Manager at Talkdesk. He holds a PhD degree in Electrical Engineering applied to High-Performance Computing at USP. Previous professional experience as Senior Software Engineer at TPN Labs in projects related to Oil & Gas companies in Brazil.

ÍCARO A. FONSECA is Adjunct Associate Professor at NTNU, where he carries research in digital systems integration, data management, and interoperability applied to the maritime sector. He also works at Hecla Emissions Management, developing solutions to assist shipowners and charterers comply with EU ETS regulations.

HENRIQUE M. GASPAR is a professor at the Department of Ocean Operations and Civil Engineering, NTNU. He coordinates the Ship Design and Operation Lab at NTNU in Ålesund. Education consists of a PhD degree in Marine Engineering at the NTNU, with research collaboration at UCL (UK) and MIT (USA). Previous professional experience as Senior Consultant at DNV (Norway) and in Oil & Gas in Brazil.