Otto-von-Guericke University Magdeburg

Department of Computer Science
Institute for Simulation and Graphics

Master Thesis

DRACO DataMaster: A Metadata-Driven Approach Utilizing Ontologies and Knowledge Graphs for the Laser Particle Acceleration

Author:
Tathagata Ghosh

February 18, 2025

Advisers:

Supervisor
Prof. Dr.-Ing. Bernhard Preim

Department of Computer Science
Otto-von-Guericke University
Universitätsplatz 2
39106 Magdeburg, Germany

Supervisor
Dr. Oliver Knodel

Department of Information Services and Computing
Helmholtz-Zentrum Dresden-Rossendorf
Bautzner Landstraße 400
01328 Dresden, Germany

Ghosh, Tathagata:
DRACO DataMaster: A Metadata-Driven Approach Utilizing Ontologies and Knowledge Graphs for the Laser Particle Acceleration
Master Thesis, Otto-von-Guericke University
Magdeburg, 2025.

Contents

Abstract

DRACO (Dresden laser acceleration source) [15] is a state-of-the-art high-power ultra-short pulse laser experiment at the Helmholtz-Zentrum Dresden-Rossendorf (HZDR), dedicated and optimized for investigating relativistic laser-plasma physics. The proposed master thesis focuses on developing a DRACO DataMaster extension that would enable advanced data handling for the DRACO experiment at HZDR. The data generated in experimental runs at the DRACO experiment comprises integer, double, float, and array values stored in tabular form in the DRACO internal database. The DRACO DATAMASTER establishes an automated pipeline for creating knowledge graphs from unsorted tabular data that will be enriched with metadata in compliance with ontologies tailored for DRACO experiments. This approach, aligned with FAIR principles (Findable, Accessible, Interoperable, and Reusable) [58], would enable deeper scientific insight and decision-making by enhancing data integration, structuring, and visualization. The proposed pipeline would significantly increase the efficiency and effectiveness of the data-driven research at HZDR, advancing experimental endeavors by providing a robust toolset for scientists.

Acknowledgements

I am really indebted to all the people who helped me during the journey of this thesis.

Above all, I would like to thank Prof. Dr.-Ing. Bernhard Preim for the supervision at the university. His advice, savvy comments, and motivation were of utmost importance in directing and influencing this thesis in style and substance.

I would also like to express my honest appreciation to Dr. Oliver Knodel for offering me such a great opportunity to perform my thesis in this modern and stimulating research atmosphere. His mentorship and trust mean a lot to me academically.

I would like to take this opportunity to express a special word of thanks to Mr. Mani Lokamani and Mr. Fredo Erxleben, who made themselves available to me with constant support during this work. Their expertise, patience, and readiness to help at each stage enabled the successful completion of this thesis.

I am equally indebted to my Abhigna Domakonda, Prudhvi, Niranjan, Amey, Dileep, Hemanth, Tarun, Akash, Anish, and Damnpreet Singh Walia for their continuous support, encouragement, understanding, and motivation. Indeed, their emotional support has been a source of strength for me during difficult times.

Finally, I owe immense gratitude to my parents, Mr. Prabir Ghosh and Mrs. Nandita Ghosh, and my brother Ramsai Chigurapathi, for their unconditional love, understanding, and encouragement. Their emotional and moral support has been my foundation and source of strength throughout my academic journey.

To all who directly or indirectly contributed to this work, thanks for being part of my academic journey.

1 Introduction and Motivation

1.1 Laser Particle Acceleration

Laser-driven particle acceleration is a ground-breaking approach, offering a transformative alternative to traditional accelerator technologies in high-energy physics. The technique uses an electric field associated with an electron plasma wave or other high-gradient plasma structures to accelerate charged particles, such as electrons or ions, efficiently [16]. The foundation concepts of plasma acceleration were proposed initially by Toshiki Tajima and John M. Dawson of UCLA in 1979 [54], while Chandrashekhar J. Joshi played a pioneering role in devising the initial experimental design of the wakefield accelerator, demonstrating the feasibility of plasma wakefield acceleration (PWFA) and its potential for high-gradient acceleration [34, 33].

Nobel laureates Donna Strickland and Gérard Mourou achieved a significant milestone in the late 1980s by developing the Chirped Pulse Amplification (CPA) [52], which revolutionized laser technology, enabling the generation of ultra-intense laser pulses and paving the way for compact and efficient particle accelerators, addressing the growing demand for versatile and cost-effective solutions in research and industrial domains.

Today's laser-driven particle accelerators have diverse applications that have extended beyond fundamental physics. Some of the impacted domains where they are being used are:

- **Medical Science:** Advancements in cancer therapy via targeted treatments that focus on minimal damage to the healthy tissues around them using a laser-accelerated proton beam. This offers precise and

less invasive treatment options with higher accuracy and efficiency [40].

- **Material Science:** Facilitating detailed microscopic analysis of material properties through sophisticated plasma-material interaction studies, emphasizing understanding the effects of low-energy plasma on material surfaces to enhance the efficiency and durability of fusion reactor components [46].

- **Industry:** Advancement in laser-driven x-ray and neutron have opened new possibilities for advanced non-destructive testing and deliver unparalleled high-resolution imaging capabilities, benefitting critical industries like aerospace, nuclear energy, and manufacturing [8].

This transformative technology has opened new possibilities across various research and application domains. The compact design of laser-driven accelerators, enabled by advancements in Chirped Pulse Amplification (CPA), provides high-energy particle acceleration in a significantly reduced footprint compared to traditional methods [39, 1, 26]. These accelerators offer a versatile solution for scientific research and present cost-effective and scalable applications in medicine, materials science, and industry.

The Development of CPA laid the foundation for developing compact accelerators and the future development of high-power laser facilities capable of achieving the petawatt-scale pulses required for relativistic plasma research. The early insights into the behavior of plasma under extreme conditions were provided by experimental platforms such as the Nova Petawatt Laser at Lawrence Livermore National Laboratory[47]. Subsequently, advancements in the field were carried out by studies on ion acceleration, laser-driven X-rays, and plasma-based light sources by Vulcan Petawatt laser in the UK and Texas Petawatt Laser [11].

1.2 DRACO Experiment

Building upon this foundation, DRACO (Dresden Laser Acceleration Source) represents one of the next-generation high-power ultra-short pulse laser

systems. At HZDR (Helmholtz-Zentrum Dresden-Rossendorf), DRACO is a state-of-the-art platform to explore and optimize relativistic cutting-edge capabilities. While retraining key features of its predecessors in the design, it also incorporated cutting-edge capabilities, which include advanced multi-beam synchronization, adaptive optics, and real-time diagnostics.

DRACO is a result of a decade of innovation, combining the relentless efforts of pioneers in high-energy laser-plasma physics research and proof of their scientific achievement. Its optimization allows fundamental scientific investigations and helps explore potential solutions in applications of laser-driven particle acceleration across industries and disciplines.

Key Features of DRACO

DRACO is built upon the Amplitude Technologies Pulsar platform, which contains a Ti: Sapphire-based laser system to achieve high temporal pulse contrast and beam quality using chirped pulse amplification (CPA). The system supports:

- **Beam Configurations:** Dual-beam architecture delivering:
  - **150 TW beam:** Producing 4.5 J of energy in 30 fs for high-precision experiments.
  - **Petawatt-class beam:** Generating up to 45 J of energy in 30 fs, enabling investigations into extreme plasma conditions.
- **Experimental Versatility:** The system is optimized for various focal lengths and target density conditions, with beams guided to specific experimental areas for tailored research applications.
- **Advanced Diagnostics:** Real-time monitoring of beam parameters, adaptive optics for superior focus, and on-shot diagnostics ensure experimental precision and reliability.

Advanced Experimental Design and Diagnostics

DRACO's experimental setup is optimized for high-intensity laser-plasma interactions by integrating diagnostics and experimental flexibility in its design, ensuring precision and adaptability for various research needs.

Target Holder for Precision Alignment: The target holder, shown in Figure 1.1, is a critical component of the experimental setup. The maximum capacity of a holder is 70 titanium foil targets, all of which are aligned to get consistent laser focus and interaction quality while minimizing alignment errors.

Figure 1.1: Precision target holder for experimental alignment.

Scintillator Detector for Proton Diagnostics: A vital diagnostic tool in DRACO's experimental setup is the scintillator detector, illustrated in Figure 1.2. The system comprises ten scintillator layers with high-resolution **imaging systems** responsible for capturing energy-resolved proton beam profiles. To ensure accurate and detailed beam diagnostics, each layer is positioned to deliver spatial and energy data across a range of proton energies.

Experimental Chamber and Beam Path: The DRACO experimental chamber, as shown in Figure 1.3, highlights the pathway of the laser beams and the arrangement of key diagnostic components. Off-axis parabolic (OAP) mirrors focus the laser beams onto the target with high precision. Multiple CCD imaging systems provide real-time feedback on focal spot alignment and target imaging, ensuring the experimental setup remains opti-

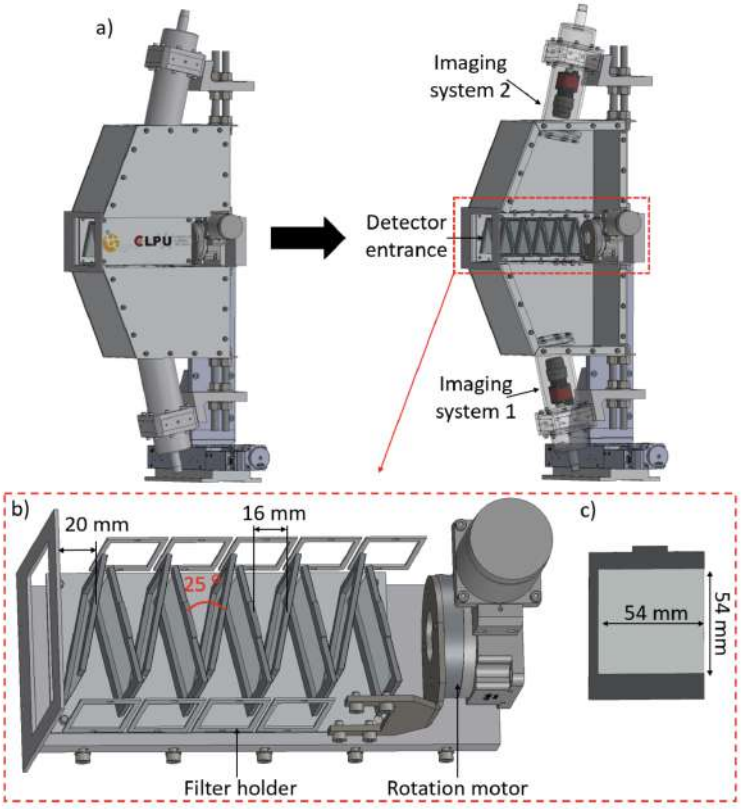

Figure 1.2: Scintillator detector system for energy-resolved proton beam diagnostics.

mized throughout the run. This sophisticated design supports simultaneous data acquisition and analysis using various diagnostic tools.

Data-Driven Innovation: Every experimental run generates a wealth of unstructured data, including high-resolution images, numerical parameters, and diagnostic results. Managing and analyzing this data requires robust storage, retrieval, and visualization systems. DRACO's ability to provide high-quality diagnostic data lays the foundation for advanced techniques such as ontology-based Data organization and knowledge graph creation for future integration with machine learning systems.

This sophisticated setup enables DRACO to support a wide range of fundamental and applied research, from exploring plasma dynamics to developing innovative applications of laser-driven particle acceleration across various industries.

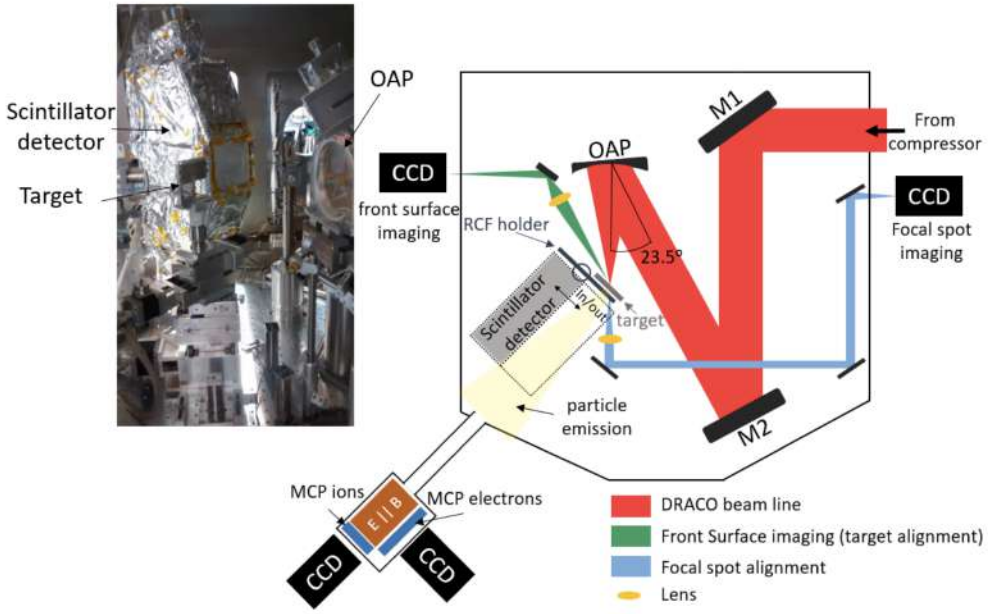

Figure 1.3: DRACO experimental setup with laser path and diagnostics.

1.3 Motivation for Advanced Data Management at DRACO

Despite the advancement in the capabilities of DRACO over the last few years, the wealth of the generated data is not properly utilized and discovered, which poses a significant challenge in the acquisition, integration, and analysis of complex data information-knowledge.

Rather than using conventional data management methods, we planned on using knowledge graphs and ontologies to understand the relationships and patterns between various parameters of the experiment, which will facilitate the goal of data-driven research. The knowledge graph, also known as a semantic network illustrates work and complex relationships between different real-world entities like objects and events, which, in our case, is perfect for illustrating different relationships between the parameters of a typical Data set and the DRACO experiment's metadata.

The ontologies, on the other hand, will define a set of terms and relationships between different entities, which in turn helps the beamline scientists at HZDR or visiting scientists to understand how they are related to each other and is one of the most important factors in the knowledge

graph. Without the knowledge from the derived ontologies, it will be just a structured graph joining various entities with each other without any sense of purpose or requirement, which explains the importance of ontologies in our system.

As mentioned, the past Data from the experiment was stored as unsorted tabular Data in an obsolete database with minimum access and oversight. This is a significant challenge as the experimental Data needs to be Findable, Accessible, Interoperable, and Reusable per the FAIR principles for laser-plasma experiments [58].

1.4 Aim of this thesis

This thesis aims to interlink the information, make it explorable, and even discover the knowledge that was previously not appropriately handled to generate new insights. This thesis identifies and investigates challenges in the existing Data acquisition and management infrastructure that are addressed by the proposed pipeline in a way that improves their efficiency and effectiveness. The final goal, as explained above, is to develop a robust Data acquisition and analysis pipeline focusing primarily on the DRACO experiment at HZDR, which will, in turn, support a visualization tool for Data exploration.

1.5 Research Questions

This thesis explores the visualization and ontology-driven management of experimental Data in DRACO. Specifically, we address the following key research questions:

- **RQ1:** How can knowledge graphs enhance the interpretability of complex experimental datasets?
- **RQ2:** What role does interactive visualization play in anomaly detection and validation of experimental results?
- **RQ3:** How can an ontology-based framework improve Data integration, retrieval, and interoperability for large-scale experiments?

1.6 Structure of this thesis

To address the issue we have on our hand, we have broken down our proposed system into three individual stages, defined as follows:

- **Data Acquisition Pipeline:** The Data acquired from the experiment will go through preprocessing to ensure it is consistent and compatible, which will then be integrated into a unified structure for the next stages.
- **Data Structure:** A structured graph is derived for the structured Data to visualize the relationship within different parameters, which will help in pattern recognition. Ontologies are developed based on recognition. Patterns and inputs from the researchers will determine formal definitions of the entities and their relationships.
- **Data Analysis:** The ontologies will lay a foundation for constructing a knowledge graph that will enable advanced querying and be a powerful visualization tool for interpreting the Data. Hierarchical clustering groups entities with similar relationships to provide a more insightful visualization.

We are planning to store the structured Data in a specialized database compatible with the Data and related metadata, which can handle and provide support for the diverse types of Data we are dealing with in the experiment.

2
Literature Review

2.1 Introduction

The development of interactive and scalable visualization systems has become a crucial aspect of modern scientific research, particularly in fields that rely on high-dimensional experimental data. Conventional approaches, such as tabular data representations, often fail to adequately capture the intricate relationships within datasets, limiting researchers' ability to gain insights. To address these challenges, knowledge graphs, force-directed layouts, visual analytics, and accessibility-driven UI design have emerged as essential methodologies.

This chapter reviews key literature in knowledge graph visualization, UX principles, perceptual design, and scientific data accessibility, explaining how these works influenced the design and implementation of the DRACO Datamaster system.

2.2 Knowledge Graphs and Semantic Data Representation

Traditional data storage methods, such as relational databases, often lack the flexibility and expressiveness required to model complex relationships in experimental datasets. To address this limitation, knowledge graphs have been widely adopted in fields such as computational science and semantic web research.

Hamilton et al. (2017) introduced a graph-based representation learning framework, demonstrating how structured relationships in large datasets

can reveal hidden patterns [22]. This research supports DRACO's use of a graph-based approach for modeling experiment-device relationships.

Noy et al. (2001) emphasized the role of ontologies in knowledge graphs, arguing that structured data representation enhances machine readability and data interoperability [44]. DRACO incorporates these principles by ensuring that its graph entities are semantically structured.

Bizer et al. (2023) further refined this idea by demonstrating how linked data principles enable efficient querying and reasoning over large datasets [7]. This principle is reflected in DRACO's backend, which integrates knowledge graphs with semantic search functionalities.

How This Influences DRACO:

- **Structured Knowledge Representation:** The graph-based approach allows hierarchical relationships to be naturally embedded.
- **Semantic Integration:** Ontology-based modeling ensures interoperability with external research databases.
- **Efficient Data Retrieval:** Enables querying of experimental results based on entity relationships.

2.3 Graph Visualization and Force-Directed Layouts

Large-scale network visualizations often suffer from node clutter, overlapping edges, and computational inefficiencies. To address these challenges, force-directed graph layouts have been widely used for arranging nodes in an intuitive and readable manner.

Fruchterman and Reingold (1991) introduced an early approach to force-directed graph layouts, which organizes nodes based on attractive and repulsive forces [19]. This method is fundamental to DRACO's knowledge graph visualization.

However, standard force-directed algorithms become computationally expensive for large datasets ($\mathcal{O}(N^2)$). To improve efficiency, **Barnes and Hut (1986)** proposed a quadtree-based approximation method, reducing complexity to $\mathcal{O}(N \log N)$ [4]. DRACO integrates this approach to ensure real-time visualization of thousands of nodes.

Jacomy et al. (2014) further improved force-directed layouts with ForceAtlas2, which enhances clustering and provides better graph stability [29]. DRACO incorporates these principles to ensure that the knowledge graph dynamically self-organizes while maintaining readability.

How This Influences DRACO:

- **Optimized Graph Layout:** The Barnes-Hut method prevents node clutter, ensuring clear visualization.
- **Hierarchical Organization:** The layout naturally groups related entities, enhancing interpretability.
- **Real-Time Updates:** Nodes adjust dynamically based on user queries and selections.

2.4 Colorblind Accessibility and Perceptual Design

Color choice in scientific visualization significantly impacts data interpretability and accessibility, especially for users with color vision deficiencies (CVD).

Brewer (1994) developed one of the first standardized approaches to perceptual color mapping, ensuring that color differences remain distinguishable across datasets [9]. DRACO applies these principles to prevent misinterpretation in its visualization components.

Jenny and Kelso (2007) analyzed how color selection impacts accessibility, recommending palettes that work effectively for all vision types [32]. These insights directly influenced DRACO's colorblind-friendly palette choices.

Tol (2021) introduced Paul Tol's muted colorblind-safe palette, which DRACO incorporates to ensure visual accessibility across all users [55].

Katsnelson (2021) highlighted how improper color usage leads to scientific misinterpretation, reinforcing the need for careful color palette selection [31].

How This Influences DRACO:

- **Inclusive Visualization:** Ensures accessibility for researchers with color vision deficiencies.

- **Optimized Contrast:** High-contrast color schemes improve readability.

- **Consistent Color Mapping:** Experiment-device relationships are visually distinguished.

2.5 User Experience and Interactive Visualization

Shneiderman (1996) introduced the Visual Information-Seeking Mantra: *"Overview first, zoom and filter, then details on demand"* [49]. This directly influenced DRACO's interactive filtering and zoom-based navigation.

Heer and Shneiderman (2012) demonstrated that interactivity enhances data comprehension, justifying DRACO's dynamic knowledge graph interactions [25].

Munzner (2014) provided heuristics for structuring complex visual data, ensuring that DRACO's graph layouts follow perceptual best practices [42].

Ware (2019) discussed cognitive overload in visual interfaces, reinforcing the need for clear hierarchy and structured navigation in DRACO [56].

How This Influences DRACO:

- **Guided Exploration:** Users start with an overview and progressively zoom into details.

- **Reduced Cognitive Load:** Filters and highlights improve focus on key relationships.

- **Optimized User Workflow:** Interactive elements enhance engagement and usability.

2.6 Summary of Literature and DRACO's Unique Contribution

Study	**Focus**	**Impact on DRACO**
Hamilton et al. (2017)	Knowledge Graphs	Supports graph-based data modeling [22].
Fruchterman & Reingold (1991)	Graph Layouts	Established force-directed placement principles [19].
Brewer (1994)	Color Theory	Defined accessibility-based color schemes [9].
Shneiderman (1996)	UX Design	Developed the interactive exploration model [49].
Ware (2019)	Cognitive Load	Provided strategies for reducing overload [56].

Table 2.1: Comparison of Key Literature and Its Influence on DRACO

3

DRACO DataOps – Managing Experimental Data

3.1 Introduction to the DRACO DB

A typical Draco experiment creates a wealth of experimental data; most of this Data is from different sources and has different types of datatypes and values with different meanings as the experimental setup has different types of devices like cameras, spectrometers some of which generate numerical data while the others provide raw images. These different types of data are stored in a non-persistent SQL-based DRACO-DB. They are stored in different datasets based on the data type of values for each measurement value.

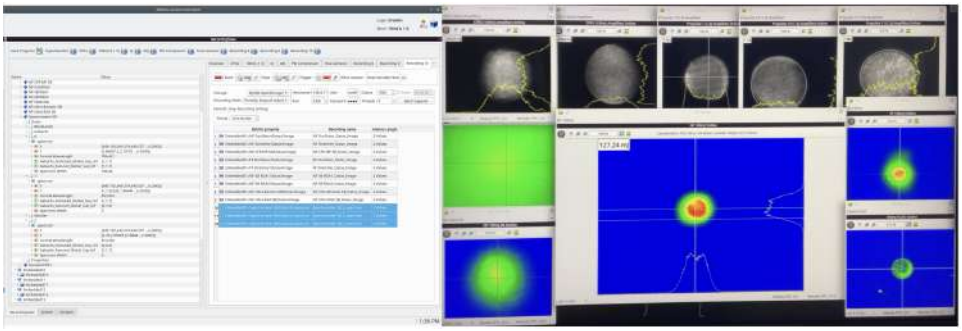

Figure 3.1: Screenshot from the DRACO control room.

The present DRACO-DB has a major flaw; it was neither designed nor developed to handle a massive amount of data nor had any documentation to rectify any problem that may arise in the database further. As time passed, the amount and experimental parameters kept evolving, but the problem with the database remained, which led to the present-day chal-

lenges where the database only has the capacity to handle two experimental run of data and needs a cronjob every month to clear space for the next batch of data.

The DRACO control room has multiple visualizations, each one dedicated to an individual measurement. As shown in Figure 3.1, the left side shows the software where different input parameters for the experiment are being given, whereas the right side image shows the individual images being recorded and the energy graph.

The data in the DRACO-DB, as mentioned, is stored in four main datasets, as shown in Fig. 2.2 till Fig. 2.5, with the identifiers:

- **RUN_DATA_ARRAY:** The dataset consists of all the measurements where the value is informed of the array datatype.

run_id	tstamp	shot_counter_id	origin_id	history_index	data
1	2024-09-02 15:28:11.915125	40559880	0000-00000032-00000000-00000001000000500000028	2	[BLOB - 3 B]
1	2024-09-02 15:28:11.915125	40559880	0000-0000003c-00000000-00000001000000500000b2	2	[BLOB - 16 B]
1	2024-09-02 15:28:11.915125	40559881	0000-0000003c-00000000-00000001000000100000098	0	[BLOB - 16.0 KiB]
1	2024-09-02 15:28:11.915125	40559881	0000-0000003c-00000000-00000001000000100000099	0	[BLOB - 16.0 KiB]
1	2024-09-02 15:28:11.915125	40559881	0000-00000032-00000000-0000000100000010000009b	0	[BLOB - 3 B]
1	2024-09-02 15:28:11.915125	40559881	0000-00000032-00000000-0000000100000010000009c	0	[BLOB - 3 B]
1	2024-09-02 15:28:11.915125	40559881	0000-00000065-00000000-00000001000000500000014	1	[BLOB - 316.5 KiB]
1	2024-09-02 15:28:11.915125	40559878	0000-0000003c-00000000-00000001000000100000098	1	[BLOB - 16.0 KiB]
1	2024-09-02 15:28:11.915125	40559878	0000-00000032-00000000-0000000100000010000009b	1	[BLOB - 3 B]
1	2024-09-02 15:28:11.915125	40559878	0000-00000032-00000000-0000000100000010000009c	1	[BLOB - 3 B]
1	2024-09-02 15:28:11.915125	40559878	0000-0000003c-00000000-00000001000000100000099	1	[BLOB - 16.0 KiB]
1	2024-09-02 15:28:11.915125	40559880	0000-00000065-00000000-00000001000000500000014	2	[BLOB - 316.5 KiB]
1	2024-09-02 15:28:11.915125	40559875	0000-0000003c-00000000-00000001000000100000098	2	[BLOB - 16.0 KiB]
1	2024-09-02 15:28:11.915125	40559875	0000-0000003c-00000000-00000001000000100000099	2	[BLOB - 16.0 KiB]
1	2024-09-02 15:28:11.915125	40559875	0000-00000032-00000000-0000000100000010000009b	2	[BLOB - 3 B]
1	2024-09-02 15:28:11.915125	40559875	0000-00000032-00000000-0000000100000010000009c	2	[BLOB - 3 B]
1	2024-09-02 15:28:11.915125	40559880	0000-0000003c-00000000-00000001000000100000076	0	[BLOB - 16.0 KiB]
1	2024-09-02 15:28:11.915125	40559880	0000-0000003c-00000000-00000001000000100000075	0	[BLOB - 16.0 KiB]
1	2024-09-02 15:28:11.915125	40559880	0000-00000032-00000000-00000001000000100000078	0	[BLOB - 3 B]
1	2024-09-02 15:28:11.915125	40559880	0000-00000032-00000000-00000001000000100000079	0	[BLOB - 3 B]
1	2024-09-02 15:28:11.915125	40559877	0000-0000003c-00000000-00000001000000100000075	1	[BLOB - 16.0 KiB]

Figure 3.2: RUN_DATA_ARRAY Dataset.

- **ORIGINID:** The dataset includes the name of the measurement, device identifier, and unique identifier for each measurement of each Device.

run_id	origin_id	datatype	top_origin_id	history_len	name	path
1	0000-00000065-00000000-00000040000002100000014	101	0000-00000065-00000000-00000040000002100000014	2	Propulse 1 (1.5J Amplifier)_Datas_Image	UNRESOLVED (33/Datas/Image)
1	0000-00000032-00000000-00000040000002100000015	50	0000-00000065-00000000-00000040000002100000014	NULL	Image.Is Object Visible	NULL
1	0000-0000003c-00000000-00000040000002100000016	60	0000-00000065-00000000-00000040000002100000014	NULL	Image.EnergyCentroid	NULL
1	0000-0000003c-00000000-00000040000002100000018	60	0000-00000065-00000000-00000040000002100000014	NULL	Image.HProfileLine.X	NULL
1	0000-00000034-00000000-00000040000002100000019	52	0000-00000065-00000000-00000040000002100000014	NULL	Image.HProfileLine.Y	NULL
1	0000-00000066-00000000-00000040000002100000017	102	0000-00000065-00000000-00000040000002100000014	NULL	Image.HProfileLine	NULL
1	0000-0000003c-00000000-0000004000000210000001b	60	0000-00000065-00000000-00000040000002100000014	NULL	Image.VProfileLine.X	NULL
1	0000-00000034-00000000-0000004000000210000001c	52	0000-00000065-00000000-00000040000002100000014	NULL	Image.VProfileLine.Y	NULL
1	0000-00000066-00000000-0000004000000210000001a	102	0000-00000065-00000000-00000040000002100000014	NULL	Image.VProfileLine	NULL
1	0000-0000003b-00000000-0000004000000210000001d	59	0000-00000065-00000000-00000040000002100000014	NULL	Image.GeometricCentroid	NULL
1	0000-0000003c-00000000-0000004000000210000001f	60	0000-00000065-00000000-00000040000002100000014	NULL	Image.Pointing Stability Values.Reference Pointing...	NULL
1	0000-0000003c-00000000-00000040000002100000020	60	0000-00000065-00000000-00000040000002100000014	NULL	Image.Pointing Stability Values.Vertical Pointing ...	NULL
1	0000-0000003c-00000000-00000040000002100000021	60	0000-00000065-00000000-00000040000002100000014	NULL	Image.Pointing Stability Values.Horizontal Pointin...	NULL
1	0000-0000003c-00000000-00000040000002100000022	60	0000-00000065-00000000-00000040000002100000014	NULL	Image.Pointing Stability Values.Total Pointing Sta...	NULL
1	0000-00000032-00000000-00000040000002100000023	50	0000-00000065-00000000-00000040000002100000014	NULL	Image.Pointing Stability Values.Timer OVer Pointin...	NULL
1	0000-00000032-00000000-0000004000000210000001e	50	0000-00000065-00000000-00000040000002100000014	NULL	Image.Pointing Stability Values	NULL
1	0000-0000003c-00000000-00000040000002100000025	60	0000-00000065-00000000-00000040000002100000014	NULL	Image.Energy or Power.Energy or Power Value	NULL
1	0000-00000032-00000000-00000040000002100000026	50	0000-00000065-00000000-00000040000002100000014	NULL	Image.Energy or Power.Energy or Power Timer Over	NULL
1	0000-00000034-00000000-00000040000002100000027	52	0000-00000065-00000000-00000040000002100000014	NULL	Image.Energy or Power.Energy or Power Unit	NULL
1	0000-00000034-00000000-00000040000002100000024	52	0000-00000065-00000000-00000040000002100000014	NULL	Image.Energy or Power	NULL
1	0000-00000032-00000000-00000040000002100000029	50	0000-00000065-00000000-00000040000002100000014	NULL	Image.Gabarits Activated.Gabarits Secured	NULL
1	0000-00000032-00000000-00000040000002100000028	50	0000-00000065-00000000-00000040000002100000014	NULL	Image.Gabarits Activated	NULL
1	0000-0000003c-00000000-0000004000000210000002a	60	0000-00000065-00000000-00000040000002100000014	NULL	Image.RealTimeFrameRate	NULL
1	0000-0000003c-00000000-00000040000002100000062	60	0000-00000065-00000000-00000040000002100000014	NULL	Image.Profiles Widths H and V	NULL
1	0000-00000065-00000000-00000040000002200000014	101	0000-00000065-00000000-00000040000002200000014	2	Propulse 10 (1.5J Amplifier)_Datas_Image	UNRESOLVED (34/Datas/Image)

Figure 3.3: ORIGINID Dataset.

- **RUN_DATA_DOUBLE:** The dataset consists of all the measurements where the value is in the form of a double datatype.

run_id	tstamp	shot_counter_id	origin_id 1	history_index	data
1	2023-11-27 17:07:40.078402	36501	0000-0000003c-00000000-0000000400000022000000 20	0	0
1	2023-11-27 17:07:40.178430	36502	0000-0000003c-00000000-0000000400000022000000 20	0	0
1	2023-11-27 17:07:39.778484	36498	0000-0000003c-00000000-0000000400000022000000 20	0	0
1	2023-11-27 17:07:39.678409	36497	0000-0000003c-00000000-0000000400000022000000 20	0	0
1	2023-11-27 17:07:39.478397	36495	0000-0000003c-00000000-0000000400000022000000 20	0	0
1	2023-11-27 17:07:39.878473	36499	0000-0000003c-00000000-0000000400000022000000 20	0	0
1	2023-11-27 17:07:39.878526	36499	0000-0000003c-00000000-000000040000002100000 02a	0	10.009099181073704
1	2023-11-27 17:07:39.478471	36495	0000-0000003c-00000000-000000040000002100000 02a	0	10
1	2023-11-27 17:07:40.478485	36505	0000-0000003c-00000000-000000040000002100000 02a	0	10.009099181073704
1	2023-11-27 17:07:39.578476	36496	0000-0000003c-00000000-000000040000002100000 02a	0	10
1	2023-11-27 17:07:40.278534	36503	0000-0000003c-00000000-000000040000002100000 02a	0	10.009099181073704
1	2023-11-27 17:07:40.378465	36504	0000-0000003c-00000000-000000040000002100000 02a	0	10.009099181073704
1	2023-11-27 17:07:39.678483	36497	0000-0000003c-00000000-000000040000002100000 02a	0	10
1	2023-11-27 17:07:39.978453	36500	0000-0000003c-00000000-000000040000002100000 02a	0	10.009099181073704
1	2023-11-27 17:07:39.278446	36493	0000-0000003c-00000000-000000040000002100000 02a	0	10
1	2023-11-27 17:07:39.178437	36492	0000-0000003c-00000000-000000040000002100000 02a	0	10
1	2023-11-27 17:07:39.778506	36498	0000-0000003c-00000000-000000040000002100000 02a	0	10.009099181073704
1	2023-11-27 17:07:40.178451	36502	0000-0000003c-00000000-000000040000002100000 02a	0	10.009099181073704
1	2023-11-27 17:07:40.078508	36501	0000-0000003c-00000000-000000040000002100000 02a	0	10.009099181073704
1	2023-11-27 17:07:39.378476	36494	0000-0000003c-00000000-000000040000002100000 02a	0	10
1	2023-11-27 17:07:39.978453	36500	0000-0000003c-00000000-000000040000002100000 025	0	1.6264304284663254
1	2023-11-27 17:07:39.878526	36499	0000-0000003c-00000000-000000040000002100000 025	0	1.6264304284663254
1	2023-11-27 17:07:39.678483	36497	0000-0000003c-00000000-000000040000002100000 025	0	1.6264304284663254
1	2023-11-27 17:07:39.578476	36496	0000-0000003c-00000000-000000040000002100000 025	0	1.6264304284663254
1	2023-11-27 17:07:39.178437	36492	0000-0000003c-00000000-000000040000002100000 025	0	1.6264304284663254
1	2023-11-27 17:07:39.378476	36494	0000-0000003c-00000000-000000040000002100000 025	0	1.6264304284663254
1	2023-11-27 17:07:39.778506	36498	0000-0000003c-00000000-000000040000002100000 025	0	1.6264304284663254
1	2023-11-27 17:07:40.378465	36504	0000-0000003c-00000000-000000040000002100000 025	0	1.628168815945304
1	2023-11-27 17:07:40.478485	36505	0000-0000003c-00000000-000000040000002100000 025	0	1.628168815945304
1	2023-11-27 17:07:40.178451	36502	0000-0000003c-00000000-000000040000002100000 025	0	1.628168815945304
1	2023-11-27 17:07:39.478471	36495	0000-0000003c-00000000-000000040000002100000 025	0	1.6264304284663254
1	2023-11-27 17:07:40.078508	36501	0000-0000003c-00000000-000000040000002100000 025	0	1.628168815945304
1	2023-11-27 17:07:40.278534	36503	0000-0000003c-00000000-000000040000002100000 025	0	1.628168815945304
1	2023-11-27 17:07:39.278446	36493	0000-0000003c-00000000-000000040000002100000 025	0	1.6264304284663254
1	2023-11-27 17:07:39.378476	36494	0000-0000003c-00000000-000000040000002100000 022	0	0
1	2023-11-27 17:07:39.178437	36492	0000-0000003c-00000000-000000040000002100000 022	0	0
1	2023-11-27 17:07:40.178451	36502	0000-0000003c-00000000-000000040000002100000 022	0	0
1	2023-11-27 17:07:39.678483	36497	0000-0000003c-00000000-000000040000002100000 022	0	0
1	2023-11-27 17:07:39.278446	36493	0000-0000003c-00000000-000000040000002100000 022	0	0
1	2023-11-27 17:07:40.078508	36501	0000-0000003c-00000000-000000040000002100000 022	0	0
1	2023-11-27 17:07:39.478471	36495	0000-0000003c-00000000-000000040000002100000 022	0	0
1	2023-11-27 17:07:39.778506	36498	0000-0000003c-00000000-000000040000002100000 022	0	0
1	2023-11-27 17:07:40.478485	36505	0000-0000003c-00000000-000000040000002100000 022	0	0
1	2023-11-27 17:07:40.278534	36503	0000-0000003c-00000000-000000040000002100000 022	0	0
1	2023-11-27 17:07:39.978453	36500	0000-0000003c-00000000-000000040000002100000 022	0	0
1	2023-11-27 17:07:39.578476	36496	0000-0000003c-00000000-000000040000002100000 022	0	0
1	2023-11-27 17:07:39.878526	36499	0000-0000003c-00000000-000000040000002100000 022	0	0
1	2023-11-27 17:07:40.378465	36504	0000-0000003c-00000000-000000040000002100000 022	0	0
1	2023-11-27 17:07:40.478485	36505	0000-0000003c-00000000-000000040000002100000 021	0	0
1	2023-11-27 17:07:40.378465	36504	0000-0000003c-00000000-000000040000002100000 021	0	0

Figure 3.4: RUN_DATA_DOUBLE Dataset.

- **RUN_DATA_INTEGER:** The dataset consists of all the measurements where the value is informed of integer datatype.

run_id	tstamp	shot_counter_id	origin_id 1	history_index	data
1	2023-11-27 17:07:39.378476	36494	0000-00000032-00000000-000000040000002100000015	0	1
1	2023-11-27 17:07:40.378465	36504	0000-00000032-00000000-000000040000002100000015	0	1
1	2023-11-27 17:07:39.178437	36492	0000-00000032-00000000-000000040000002100000015	0	1
1	2023-11-27 17:07:40.478485	36505	0000-00000032-00000000-000000040000002100000015	0	1
1	2023-11-27 17:07:39.778506	36498	0000-00000032-00000000-000000040000002100000015	0	1
1	2023-11-27 17:07:39.878526	36499	0000-00000032-00000000-000000040000002100000015	0	1
1	2023-11-27 17:07:40.278534	36503	0000-00000032-00000000-000000040000002100000015	0	1
1	2023-11-27 17:07:39.678483	36497	0000-00000032-00000000-000000040000002100000015	0	1
1	2023-11-27 17:07:40.178451	36502	0000-00000032-00000000-000000040000002100000015	0	1
1	2023-11-27 17:07:39.578476	36496	0000-00000032-00000000-000000040000002100000015	0	1
1	2023-11-27 17:07:40.078508	36501	0000-00000032-00000000-000000040000002100000015	0	1
1	2023-11-27 17:07:39.978453	36500	0000-00000032-00000000-000000040000002100000015	0	1
1	2023-11-27 17:07:39.478471	36495	0000-00000032-00000000-000000040000002100000015	0	1
1	2023-11-27 17:07:39.278446	36493	0000-00000032-00000000-000000040000002100000015	0	1
1	2023-11-27 17:07:39.478471	36495	0000-00000032-00000000-000000040000002100000001e	0	0
1	2023-11-27 17:07:39.678483	36497	0000-00000032-00000000-000000040000002100000001e	0	0
1	2023-11-27 17:07:39.778506	36498	0000-00000032-00000000-000000040000002100000001e	0	0
1	2023-11-27 17:07:39.278446	36493	0000-00000032-00000000-000000040000002100000001e	0	0
1	2023-11-27 17:07:40.078508	36501	0000-00000032-00000000-000000040000002100000001e	0	0

Figure 3.5: RUN_DATA_INTEGER Dataset.

As described above, the database is old and has limited capacity, which leads to a Scalability issue; along with that, the DRACO DB has limited accessibility, which in turn leads to problems in the interoperability of data between different teams. To solve all of these issues, we proposed a system that will not only solve the problem related to Scalability and long-time storage of data but also be accessible to different groups as part of the experiment.

3.2 DRACO-DB dataset

To propose an effective solution for the DRACO-DB issues the researchers are seeking, we need to understand different parameters in the Data and make a system that satisfies the FAIR (Findable, Accessible, interoperable, Reusable) principle. These parameters in the Data are critical as they represent key experimental values and identifiers, which, in turn, help us structure the Data and ensure its traceability.

3.2.1 Key Parameters in DRACO-DB

1. `shot_counter_id`

This parameter acts as a unique counter for each shot or measurement recorded during an experiment.

Importance:

- Helps identify the sequence of shots taken.
- Ensures chronological order of measurements, which is critical for time-dependent analyses.

2. `origin_id`

A composite identifier containing information about the measurement event being carried out by the Device, its unique ID.

Importance:

- Links Data to specific experimental devices and configurations.

- Essential for troubleshooting and tracing Data back to its source.

3. `top_origin_id`

Represents the Device that is responsible for a particular measurement value.

Importance:

- Helps track changes or updates to measurements over time.
- Ensures Data integrity and version control.

4. `Data`

The main column holds the measurement values. These values can belong to:

- `RUN_DATA_INTEGER`: Contains all the measurement having integer-type values.
- `RUN_DATA_DOUBLE`: Contains all the measurement having Continuous or floating-point values.
- `RUN_DATA_ARRAY`: Contains all the measurements having Array-type values that include image blobs or multi-dimensional measurements.

Importance:

- Forms the core experimental Data necessary for analysis, visualization, and further computations.

5. `tstamp`

A timestamp indicates when each measurement was recorded.

Importance:

- Allows time-based sorting and analysis.
- Facilitates synchronization of Data across multiple devices and measurements.

3.3 Proposed System Architecture: DRACO-DATAMASTER Architecture

The limitations of the existing DRACO-DB, such as limited capacity, accessibility, and interoperability, require the development of a new, robust, and modern system. The proposed system, **DRACO-DATAMASTER**, provides a solution for all these challenges not only by providing a long-term storage solution but also by structuring the Data based on a well-defined schema. Ontologies are created along knowledge graphs to properly define the experiment and the relationship between different key parameters recorded during it.

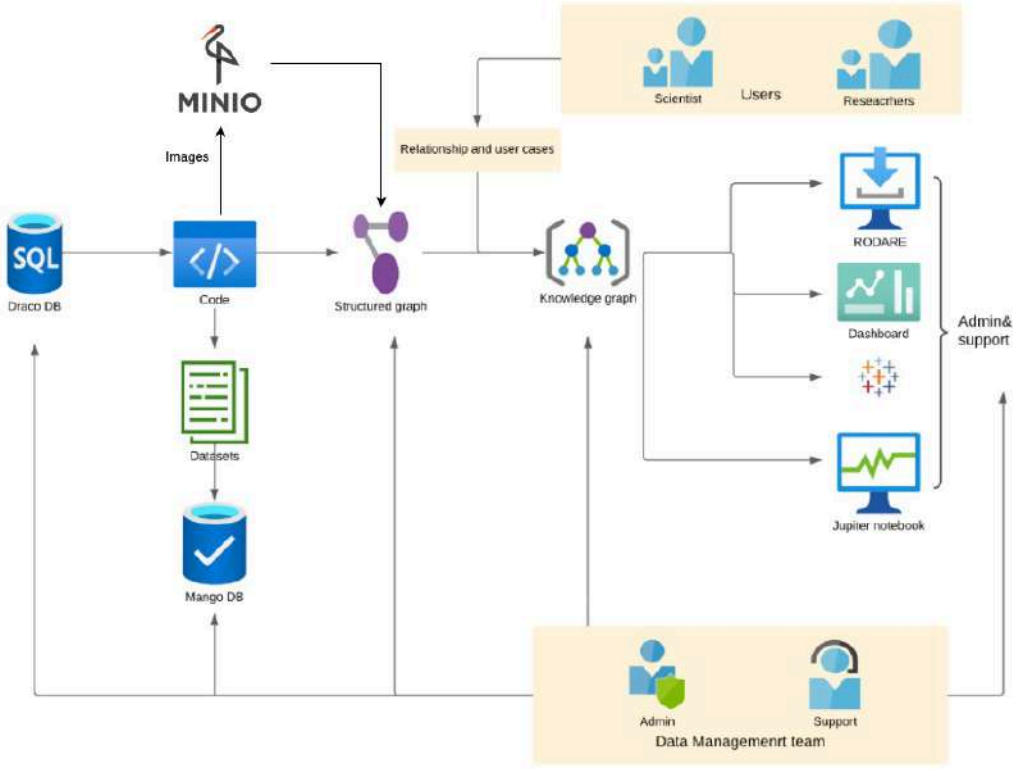

Figure 3.6: System Architecture.

3.3.1 Overview of the Architecture

The architecture, as shown in Figure 3.6, is designed by considering all the requirements related to the challenges faced by the DRACO-DB. It ensures a streamlined Data flow along with Scalability while facilitating user interactions and future development.

It comprises the following components:

1. **Draco DB (SQL-based)**: The existing SQL-based DRACO is located in the experiments control room, not part of the central IT, and our long-term storage and post-processing serve as the initial Data source. It stores experimental Data recorded from various devices like cameras and spectrometers. In the future, it will be entirely replaced by a pipeline. Our system acts as a staging database only, which will hold the Data for a short while before pushing it into the other database.

2. **Code and Structured Graph**: Currently, Data is extracted from the SQL database using a Jupyter Notebook. It is processed through scripts and transformed into various formats before being uploaded to the databases. In the future, IT plans to provide a job on our cluster or a dedicated virtual machine that will handle Data extraction on demand.

   Importance:

   - Handles reorganization and restructuring of the Data.
   - Reduces redundancy and prepares the Data for further processing based on the required use cases.

3. **Mongo DB**: A MongoDB instance stores the Data in a properly structured way based on the metadata schema we prepared. It helps enhance Scalability and quick access. We also offer backups on tape for long-term archiving of enormous amounts of Data.

   Importance:

   - Responsible for long-term storage for a large dataset.

- The Data is being structured and saved as a JSON file.
- MongoDB helps us in faster querying and retrieval of Data.

4. **Ontologies**: Based on the relationship identified between the parameters, we will create an ontology for the experiment.

   Importance:

   - Improve clarity and understanding of the relationship.
   - Supports advanced queries and insights.

5. **Knowledge Graph**: The structured graph Data is formed using the parameter Data from the experiment. When integrated with ontologies, we are able to create a knowledge graph. It is further enriched to form a knowledge graph.

   Importance:

   - A graph representing the relationship between the datasets, devices, and experiments.
   - Supports advanced Data analytics and discovery.

6. **Visualization and User Tools**: For Data analysis and visualization, various tools are integrated into our system.

   - **RODARE**: For Data publication and sharing.
   - **Dashboard**: Developed a Graphical User Interface to visualize the Data insights.
   - **Jupyter Notebook**: Allows researchers to analyze and visualize Data interactively.

3.3.2 Data Workflow

The proposed system architecture Figure 3.6 provides us with a smooth workflow for Data management:

- Experiment Data is first ingested into the **Draco DB**.
- The experimental Data is then retrieved for Preprocessing, which is followed by running scripts that derive **structured**

graph from the Data and store measurement value in **MongoDB** in the form of a JSON File while the images are stored in the MinIO DB.

- Ontologies are developed based on the identified relationships in the datasets.
- Based on the developed ontologies, we will develop the **knowledge graph**.
- Users access data and insights through **RODARE**, dashboards, and Jupyter Notebooks.

The DRACO-DATAMASTER system addresses the critical challenges of the DRACO-DB by enhancing Scalability, accessibility, and interoperability. This architecture provides the researchers with resources to efficiently manage and analyze the experimental Data for future scientific development.

3.4 Identifying Relationships and Data Integration

This section centers on identifying the relations between tables of DRACO-DB and figuring out how these relationships help us bring them together to carry out a unified analysis and understanding.

3.4.1 Identifying Relationships

DRACO-DB have multiple different datasets, as we have explained about them in section 3.1, `ORIGINID`, `RUN_DATA_INTEGER`, `RUN_DATA_DOUBLE`, and `RUN_DATA_ARRAY` stores all different types of Data from devices in them. The following relationships were identified:

1. Device-to-Measurement Relationship

In `ORIGINID` dataset each `top_origin_id` which corresponds to a device to a set of measurement value identifier `origin_id` and we get the name for that measurement from the parameter `name`. Alone it does not give us much information but all the

`origin_id` can be found in in the datasets `RUN_DATA_INTEGER`, `RUN_DATA_DOUBLE`, or `RUN_DATA_ARRAY`.

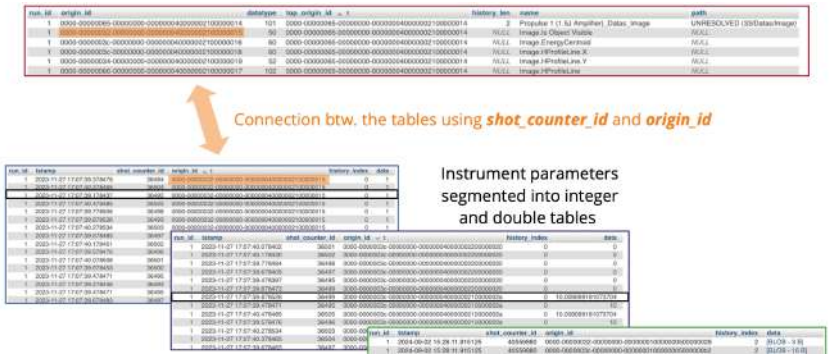

Figure 3.7: Origin_id based relationship

As shown in Figure 3.7 `origin_id`, connect different datasets and provide us with details regarding the measurements that we are working on without the `shot_counter_id` we will not be able them to map them to particular experimental settings. The `shot_counter_id` present in the other dataset hence will be able to bring all the required values together by using them.

2. Temporal Relationship

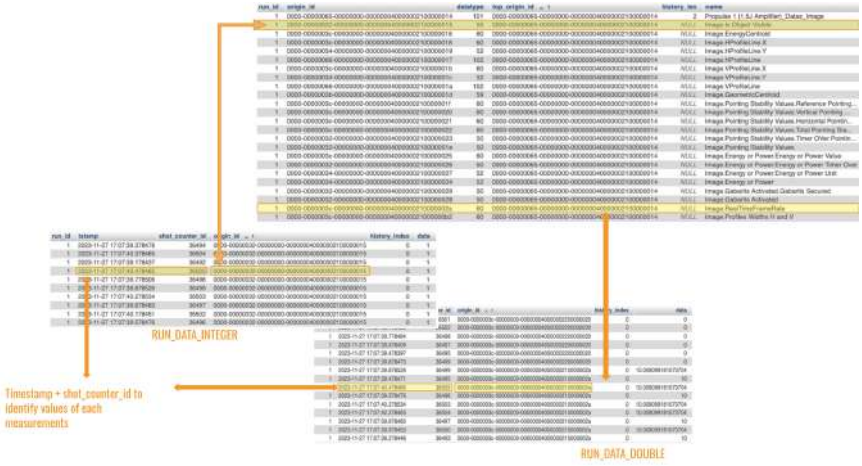

Figure 3.8: Timestamp and shot_counter_id relation with experiment

As shown in Figure 3.8 `tstamp` and `shot_counter_id` are important along with `origin_id` to identify the a exact measurement and

its value. a shot has multiple measurements taken, and each has a distinct name; based on the number of devices being used in that particular shot, we will have multiple measurement values.

`tstamp` help us in grouping together all the measurements for a shot in chronological order, helping us to understand the shot measurement progression. The grouping also helps us combine the Data from all the datasets in a proper structure while maintaining the integrity of the Data and ensuring fast retrieval.

3.4.2 Unifying the Tables

The identified relationships allow us to integrate the tables into a single structured dataset. By linking these tables using `origin_id` and `shot_counter_id`, we create a cohesive representation of the experiment Data, ensuring traceability and consistency.

To develop this, we create the following Data structure Figure 3.9; in this, we have grouped together all the parameters such as `origin_id`, `name`, `top_origin_id` that define the type of Device being used in the experiment run in a class named Device. Where as the class Measurement values group together parameter like `value` and `tstamp` corresponding to the `shot_counter_id` of the experiment. The Shot class is responsible for retrieving the `shot_counter_id` whereas the Run class for `run_id` and Experiment class for the experiment date.

3.5 JSON Structure for MongoDB and MinIO

The following subsections describe how measurement values and images are stored for Scalability and accessibility.

3.5.1 Measurement Storage in MongoDB

The structured measurement Data from the DRACO-DB is stored as JSON documents in MongoDB. The Data retrieval process extracts key information, such as measurement values, timestamps, and device metadata, which are transformed and stored in a hierarchical format for long-term access.

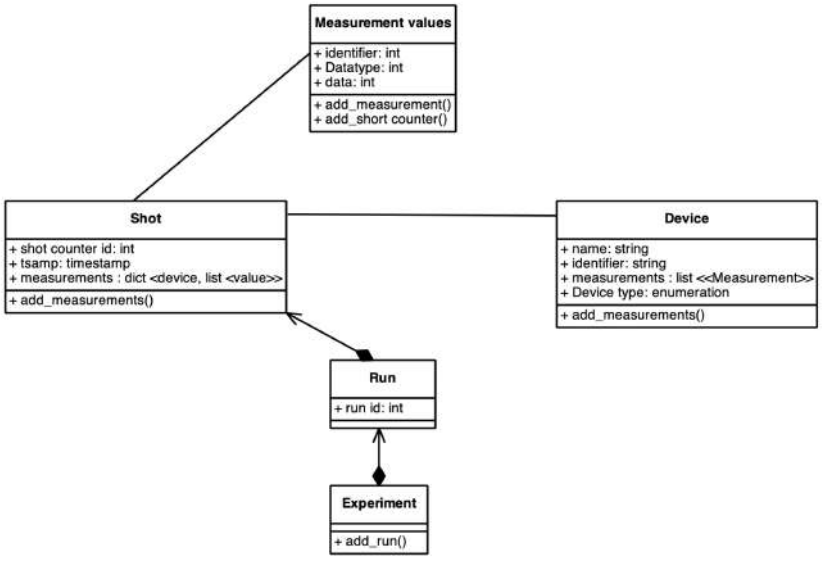

Figure 3.9: Proposed class structure of experiment

Data Retrieval from SQL Database

Measurement data is extracted from the `ORIGINID`, `RUN_DATA_INTEGER`, `RUN_DATA_DOUBLE`, and `RUN_DATA_ARRAY` datasets of the DRACO SQL database.
The SQL query for retrieving the Data is shown below:

```
SELECT o.origin_id, o.name, o.top_origin_id,
       COALESCE(rdd.shot_counter_id, rdi.shot_counter_id,
       RDA.shot_counter_id) AS shot_counter_id,
       COALESCE(rdd.tstamp, rdi.tstamp, rda.tstamp) AS timestamp,
       rdd.Data AS double_data, rdi.data AS integer_data,
       rda.data AS array_data
FROM ORIGINID o
LEFT JOIN RUN_DATA_DOUBLE rdd ON o.origin_id = rdd.origin_id
LEFT JOIN RUN_DATA_INTEGER rdi ON o.origin_id = rdi.origin_id
LEFT JOIN RUN_DATA_ARRAY rda ON o.origin_id = rda.origin_id
WHERE o.top_origin_id = '{device_identifier}'
AND o.path IS NULL;
```

This query extracts:

- `origin_id`: Unique identifier linking devices and measurements.
- `shot_counter_id`: Identifies the shot or measurement event.

- `tstamp` : Time at which the measurement was recorded.
- `data` : Measurement values, which we retrieve from `RUN_DATA_INTEGER` , `RUN_DATA_DOUBLE` , or `RUN_DATA_ARRAY` .

Decoding and Processing Data

The retrieved `array_data` column, which stores binary Data, is decoded using the Python `struct` library. The following code snippet demonstrates the decoding process:

```
import struct

def decode_binary_data(binary_data):
    """ Decode binary array data into floating-point values."""
    decoded_values = []
    for i in range(0, len(binary_data), 8):
        chunk = binary_data[i:i+8]
        if len(chunk) == 8:
            decoded_value = struct.unpack('d', chunk)[0]
            decoded_values.append(decoded_value)
    return decoded_values
```

This ensures that the extracted `array_data` is transformed into human-readable and analyzable formats before storage.

Data Structuring:

The retrieved Data is organized into a nested JSON format, enabling efficient querying and Scalability. The hierarchical structure groups measurements based on experiments, runs, shots, and devices.

JSON Schema for MongoDB

The following JSON schema illustrates how the experimental data is stored in MongoDB. The schema includes multiple nested levels for clear organization.

Example JSON Document:

```
[
  {
    "_id": "ObjectId('67360aa02e1e082bc5232b8c')",
    "experiment": [
      {
        "run_id": 1,
        "shots": [
          {
            "shot_counter_id": 36492,
            "Devices": [
              {
                "name": "0000-00000067-00000000-0000000200000000f00000097",
                "measurements": [
                  {
                    "timestamp": "2023-11-27 17:07:39.174259",
                    "data": [
                      {
                        "measurement_name": "spectrum.Y",
                        "data": [1, 1.33, 1.44, 1250.55]
                      },
                      {
                        "measurement_name": "spectrum.X",
                        "origin_id": "0000-0000003c-00000000-0000000200000000f00000098",
                        "data": [651.85, 652.03, 652.21, 652.39]
                      },
                      {
                        "measurement_name": "spectrum.Central Wavelength",
                        "origin_id": "0000-0000003c-00000000-0000000200000000f0000009a",
                        "data": 805.68
                      },
                      {
                        "measurement_name": "spectrum.Spectrum Width",
                        "origin_id": "0000-0000003c-00000000-0000000200000000f0000009d",
                        "data": 61.01
                      }
                    ]
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
  }
]
```

Folder Hierarchy and Organization

The hierarchical structure in MongoDB reflects the following organization:

- **Experiment Level:** Each document corresponds to a unique experiment.
- **Run Level:** Groups all shots belonging to a single experimental run.
- **Shot Level:** Contains measurements collected during an individual shot, identified by `shot_counter_id`.
- **Device Level:** Includes the device-specific measurements and their metadata.

Advantages:

- Ensures Scalability and efficient organization of hierarchical data.
- Simplifies querying and retrieval of measurement values for a given experiment or shot.
- Facilitates long-term storage and access for collaborative research.

3.5.2 Image Storage in MinIO

The MinIO object storage system is used to store experimental images captured during the DRACO experiments. Each image file is processed, converted to a standardized format, and uploaded with metadata to ensure **scalability**, **accessibility**, and **traceability**.

Workflow for Image Storage

The image storage workflow consists of three key stages: **Data Retrieval**, **Processing Raw Image Data**, and **File Naming and Organization**.

1. Data Retrieval Images are retrieved from the `RUN_DATA_ARRAY` table in the SQL-based DRACO database. The query specifically selects rows where the column `path IS NOT NULL`.

The following columns are extracted:

- `origin_id`: Unique identifier for the measurement.
- `top_origin_id`: Unique identifier for the device.
- `shot_counter_id`: Identifies the shot/measurement.
- `tstamp`: Timestamp of when the image was captured.
- `array_data`: Binary raw image Data.

The SQL query used for retrieval is as follows:

```
SELECT o.origin_id, o.name, o.top_origin_id,
       rda.data AS array_data, rda.shot_counter_id AS shot_counter_id,
       rda.stamp AS timestamp
FROM ORIGINID o
LEFT JOIN RUN_DATA_ARRAY rda
    ON o.origin_id = rda.origin_id
WHERE o.path IS NOT NULL
```

2. Processing Raw Image Data The raw binary Data from the `data` column in `RUN_DATA_ARRAY` dataset is decoded and converted into a standardized **TIFF format** using the `raw2tiff` tool. This process ensures compatibility for long-term storage and visualization.

The steps are as follows:

(a) Binary data is saved temporarily as a `.raw` file.

(b) The `raw2tiff` tool converts the file to `.tiff` format.

(c) Temporary files are removed after successful conversion and upload.

Code Example:

```
with open(raw_file, "wb") as f:
    f.write(array_data)
subprocess.run(f"raw2tiff -w 656 -l 494 {raw_file} {tiff_file}", shell=True)
```

3. File Naming and Organization To ensure clarity and uniqueness, images are named and stored following a structured convention:

- **File Naming Convention:**

  `<experiment_id>/<device_identifier>/<origin_id>_<shot_counter_id>_<timestamp>.tiff`

- **Folder Hierarchy:**

```
<experiment_id>/
    <device_identifier>/
        <origin_id>_<shot_counter_id>_<timestamp>.tiff
```

Fields:

- `experiment_id` : Experiment identifier (e.g., date of the experiment).
- `origin_id` : Unique identifier for the measurement values.
- `top_origin_id` : Unique identifier for the measurement device.
- `shot_counter_id` : Identifies the shot.
- `tstamp` : Timestamp indicating when the image was recorded.

Advantages of MinIO for Image Storage

The MinIO storage solution provides significant benefits for managing DRACO experiment images:

- **Scalability :** Supports large volumes of image data efficiently.
- **Accessibility:** Enables fast and efficient retrieval using metadata.
- **Traceability:** Metadata links each image to its respective experiment and measurements.
- **Standardization:** Conversion to TIFF format ensures compatibility across visualization tools.
- **Organization:** Structured file naming and folder hierarchy prevent duplication and simplify management.

Workflow Diagram

Figure 3.10 illustrates the example of an image being stored in the MinIO DB. This is the final output, from SQL data retrieval to final upload to MinIO.

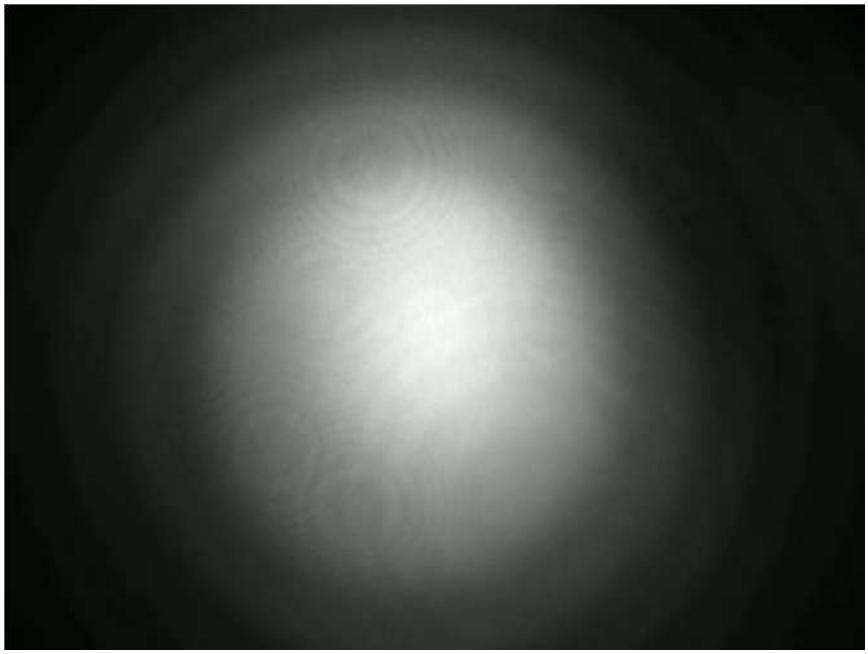

Figure 3.10: Image Storage Workflow in MinIO

3.6 Integration Pipeline

The integration pipeline combines the retrieval, transformation, and storage of experimental data. Ensure that both measurement values and images are processed efficiently and stored in MongoDB and MinIO, respectively.

The pipeline leverages Python's `ThreadPoolExecutor` for parallel processing, and robust error-handling mechanisms ensure that issues such as missing data or failed uploads do not halt the pipeline.

3.6.1 Parallel Processing

To optimize the efficiency of the data integration process, the pipeline uses Python's `ThreadPoolExecutor` for parallel execution of tasks. This allows simultaneous retrieval, processing, and storage of:

- Measurement values in MongoDB.
- Image files in MinIO.

Workflow

The following workflow outlines the process:

(a) Retrieve all unique `top_origin_id` values from the DRACO SQL database.

(b) For each `top_origin_id`, process measurements and images concurrently.

(c) Store measurement data in MongoDB and images in MinIO.

The following code snippet demonstrates how measurements and images are processed concurrently:

```
from concurrent.futures import ThreadPoolExecutor

def main():
    experiment_id = f"experiment_{datetime.now().strftime('%Y%m%d')}"
    top_origin_ids_query = "SELECT DISTINCT top_origin_id
                            FROM ORIGINAL
                            WHERE path IS NULL"
    top_origin_ids = pd.read_sql(top_origin_ids_query, db_ops.engine)['top_origin_id'].tolist()

    with ThreadPoolExecutor() as executor:
        futures = []
        for top_origin_id in top_origin_ids:
            device = Device(name=top_origin_id, identifier=top_origin_id)
            futures.append(executor.submit(
                device.get_measurements_with_shot_counter, db_ops, None
            ))
            futures.append(executor.submit(
                device.process_and_upload_images, db_ops, minio_handler, experiment_id
            ))

        for future in futures:
            future.result()
```

Key Features:

- **Concurrency:** Measurements and images for each Device are processed simultaneously, reducing execution time.
- **Scalability:** The number of worker threads can be adjusted based on hardware resources.
- **Flexibility:** Each Device runs independently, enabling better error isolation.

3.6.2 Advantages of the Pipeline

- **Efficiency:** Parallel processing reduces the total time required for data ingestion and storage.
- **Reliability:** Robust error handling ensures that the pipeline remains functional despite data or system failures.
- **Scalability:** The pipeline can handle multiple devices and large datasets concurrently.
- **Traceability:** Logs and retry mechanisms provide transparency into failed operations for troubleshooting.

3.7 Summary

This chapter introduced the DRACO DataOps framework, detailing its role in managing experimental data efficiently. We explored the structured approach used for data collection, storage, and processing, emphasizing interoperability and Scalability.

A key highlight was the use of metadata-driven ontologies to ensure consistency in experimental datasets, enabling better data retrieval and organization. The integration of a knowledge graph structure further enhances data relationships, allowing for more intuitive data exploration.

In the next chapter, we will delve deeper into the ontology-driven model used in DRACO, explaining how semantic structures are leveraged to improve data representation and accessibility.

4

Ontology-Driven Data Representation in DRACO

4.1 Introduction to Ontologies

Ontologies provide a systematic account of shared conceptual frameworks within a specific domain by offering a structured representation of knowledge that enriches reasoning, interoperability, and data integration. They transform complex datasets into semantically rich models that facilitate advanced data management and analysis by specifying entities, their characteristics, and their relationships [21]. This approach aligns closely with the core tenets of contemporary scientific data management and stewardship, especially the FAIR principles (findability, accessibility, interoperability, and reusability) [58].

In the context of the DRACO experiment, ontologies are essential for organizing and managing experimental data. They enable the domain-specific knowledge obtained during experiments by the researchers to be encoded in a machine-readable format, making the data comprehensible, reusable, and interoperable across various research teams and computational systems. By adopting ontologies, DRACO transforms complex datasets into semantically rich models, achieving semantic consistency, enhancing cross-experiment comparisons, and fostering interdisciplinary collaboration [50].

Why Ontologies?

Ontologies serve several purposes in scientific and technical domains, including addressing critical challenges in data-driven research by offering several key advantages:

- **Standardization:** Establishing standard vocabularies to describe data, processes, and entities, ensuring semantic uniformity across domains [44].
- **Data Integration:** Ontologies Bridge heterogeneous datasets by providing a shared semantic framework, enabling interoperability [7].
- **Knowledge Representation:** They capture domain-specific knowledge in a formalized, machine-readable format, fostering transparency and computational reasoning [21].
- **Query and Reasoning:** Ontologies support advanced queries and logical reasoning, enabling researchers to derive insights from structured knowledge [27].

Core Components of an Ontology

As described by [44], an ontology typically consists of the following components:

- **Classes:** Represent the fundamental concepts or entities in a domain (e.g., "Device," "Experiment").
- **Properties:** Define attributes of classes or relationships between them, such as "hasMeasurement" or "isPartOf".
- **Instances:** Represent specific examples of the classes (e.g., a particular sensor used in an experiment).
- **Axioms:** Define constraints, rules, or logical assertions that govern the relationships between classes and properties. These axioms ensure data consistency and semantic clarity

Ontologies in Data-Driven Research

Modern experiments, such as DRACO, generate vast amounts of data that require proper organization, integration, and analysis. Ontologies serve as a powerful tool to:

- Enhance the discoverability of datasets by making their structure explicit and standardized.
- Ensure semantic interoperability for cross-experiment comparisons, enabling researchers to integrate datasets from diverse sources [58]

- Facilitate the creation of knowledge graphs that support advanced data visualization and generate new insights [7][27]

By implementing ontologies, the DRACO experiment takes significant strides toward achieving these objectives, making its data ecosystem more transparent, reusable, and robust.

Evolution of Ontologies

The concept of ontologies originated in philosophy and was later adopted in artificial intelligence (AI) as a framework for knowledge representation [21] . Over time, they have been implemented in various different domains, including life science [2], engineering [44], and data management [58].

In the case of DRACO, the development and implementation of ontologies signifies a significant evolution in experimental data curation, enabling semantic annotation, structured integration, and enhanced reasoning. This approach underscores the growing importance of ontologies in addressing the challenges posed by the increasing complexity and volume of scientific data [50].

4.2 Ontology Development for DRACO

4.2.1 Purpose

The DRACO experiment ontologies were developed to provide a clear formal relationship between different entities in the experiment. The primary objective is to provide a structured framework that enhances metadata alignment and supports the generation of knowledge graphs, which further assist in visualizing, exploring, and analyzing experimental Data.

- **Hierarchical Data Representation:** Capturing nested relationships, such as the connection between runs, shots, and individual measurements.
- **Semantic Enrichment:** Enhancing experimental Data with contextual meaning, ensuring it is both human-readable and machine-interpretable.

- **Data Reusability:** Enabling Data reuse across experiments and research teams by adhering to standardized semantic models.

By providing a formalized structure, the ontology empowers researchers to efficiently query, compare, and reuse data, ultimately fostering collaboration and advancing scientific discovery.

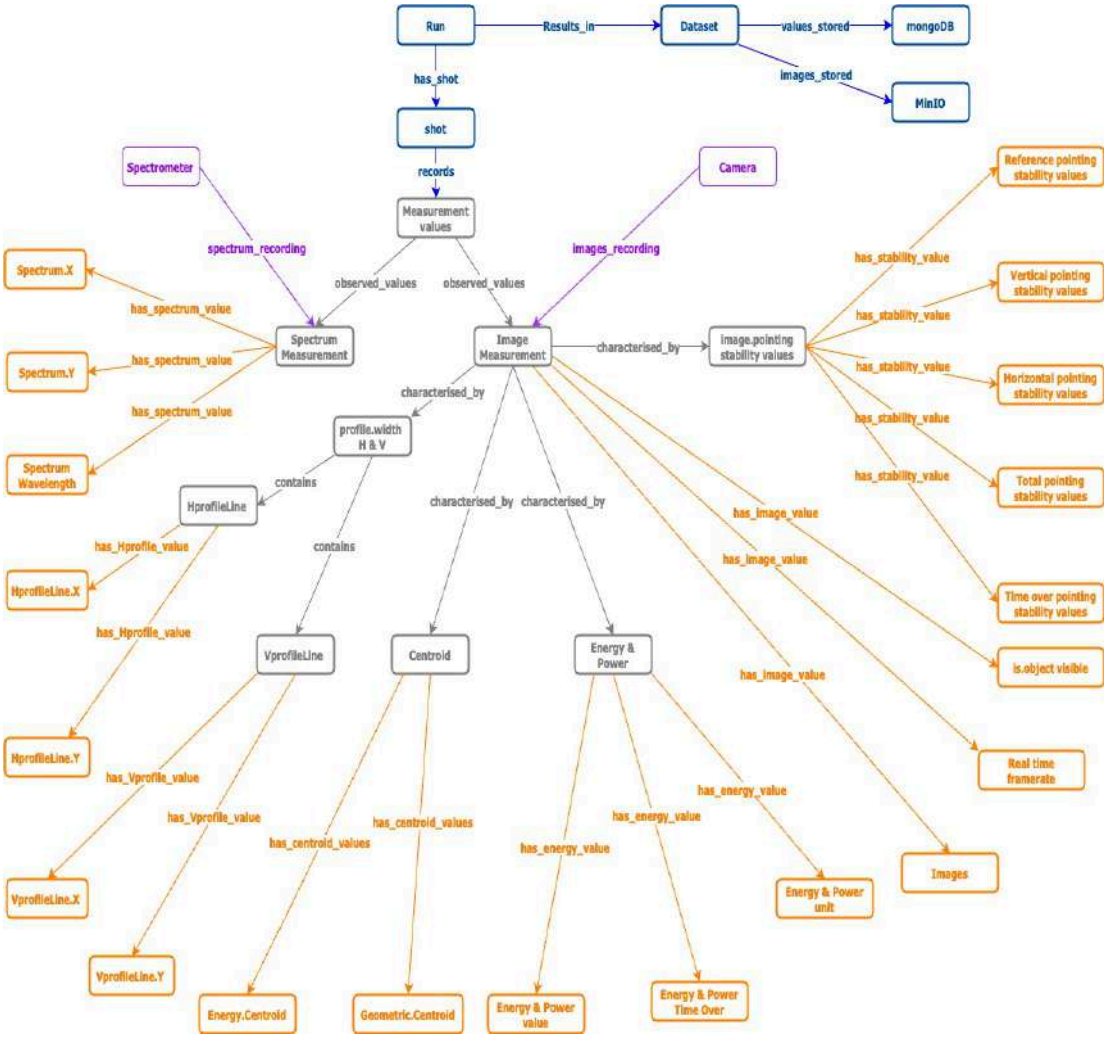

Figure 4.1: A visual representation of the ontology, showing the core classes and their relationships.

4.2.2 Design Process

Defining Classes and Hierarchies:

The hierarchical structure of the ontology, including core classes such as Devices, Measurements, Runs, and Shots, is illustrated in Figure 4.1. This diagram demonstrates how these entities and their relationships are modeled to ensure semantic clarity and interoperability.

Class	Subclass/Attribute	Description
Devices	`Spectrometer`, `Camera`	Experimental equipment responsible for recording specific types of measurements.
Measurements	`Spectrum Measurement`, `Image Measurement`	Central classes representing the data recorded by devices.
Runs	None	Represents iterative executions of the experimental setup.
Shots	None	Captures individual instances within a Run.
Derived Data	`Energy & Power`, `Centroid`	Secondary attributes calculated from measurements, supporting advanced analysis.

Table 4.1: Overview of Classes and Hierarchies

Runs and Shots:

Runs: The `Run` class contains the details of how many times the experimental setup is executed to capture measurements within a specific experiment. It serves as the overarching entity that organizes the experiment into multiple runs.

Shots: The `Shot` class contains the data of the measurement corresponding to each individual shot within an experimental run.

The relationships between ‘Run‘ and ‘Shot‘ classes are defined as follows:

Class	Attribute/Relationship	Description
`Run`	`has_shot`	Links a Run to its associated Shot instances.
`Shot`	`records`	Indicates that a Shot records measurements.

Table 4.2: Structure of Runs and Shots

Measurements:

Measurements form the central class, encapsulating both `Spectrum Measurement` and `Image Measurement`. Subclasses of measurements include:

Measurement Type	Attribute	Description
`Spectrum Measurement`	`spectrum.X`, `spectrum.Y`	Captures x and y coordinates of the spectral data.
	`Spectrum Wavelength`	Represents wavelength information within the spectrum.
	`Spectrum Width`	Indicates the width of the spectrum recorded.
`Image Measurement`	`profile.width H & V`	Horizontal and vertical profile widths of the image.
	`Centroid`	Centroids representing energy and geometry of the captured data.
	`RealTimeFrameRate`	Represents the real-time frame rate of the image capture.
	`Pointing Stability Values`	Stability metrics for horizontal, vertical, and overall pointing systems.
	`Energy or Power Value`	Representing the Energy or Power measurement of the captured data.

Table 4.3: Attributes of Measurements

Devices:

As discussed in Chapter 2, each experiment consists of multiple devices and each responsible for a broad set of measurements. The devices mostly fall into two types: spectrometers and cameras, which serve distinct purposes in data collection. In the ontology, subclasses such as `Spectrometer` and `Camera` are created to represent experimental equipment. The measurements recorded by these devices are linked to specific recording processes and are grouped together, such as `Spectrum_Measurement` and `Images_Measurement` by the `Spectrometer` and `Camera`, respectively.

Device Class	Measurement Type	Attributes
Spectrometer	Spectrum Measurement	Spectrum.X, Spectrum.Y, Spectrum Wavelength, Spectrum Width, Gabarits_Activated
Camera	Image Measurement	Profile.width H & V, EnergyCentroid, GeometricCentroid, RealTimeFrameRate

Table 4.4: Devices and Their Measurement Capabilities

Derived Data:

Derived attributes provide additional analytical insights and are calculated from raw measurements. These attributes include:

Derived Data	Attribute	Description
Energy & Power	Energy or Power Value	Represents energy or power measurements recorded.
	Energy or Power Unit	Specifies the unit of measurement (e.g., joules, watts).
	Energy or Power Unit	Specifies the unit of energy or power measurement.
Centroid	EnergyCentroid	Centroid representing the energy of the recorded data.
	GeometricCentroid	Geometric centroid of the captured data.
Pointing Stability	Total Pointing Stability	Overall pointing stability derived from horizontal and vertical metrics.
	Reference Pointing Stability Values	Contains the stability of the reference pointing system.
	Vertical Pointing Stability Values	Contains the vertical stability metrics stability.
	Horizontal Pointing Stability Values	Contains the horizontal stability metrics stability.
	Total Pointing Stability Values	Contains the overall pointing stability metrics.
	Timer OVer Pointing Stability Values	Contains the data that tracks the stability metrics over

Table 4.5: Attributes of Derived Data

4.3 Entity Relationships in the DRACO Ontology:

The ontology employs rich relationships to connect entities and define their interactions. Figure 4.1 provides a visual representation of the relationships among key ontology entities, such as the connections between `Devices` and `Measurements` through `has_measurement_value`, and between `Runs` and `Shots` via `has_shot`. Key relationships include:

4.3.1 Device Relationships:

Device relationships establish how specific equipment generates measurements. Table 4.6 outlines these associations.

Relationship	Source Class	Target Class
`images_recording`	`Camera`	`Images_Measurement`
`spectrum_recording`	`Spectrometer`	`Spectrum_Measurement`

Table 4.6: Device Relationships

4.3.2 Measurement Relationships:

Table 4.7 summarizes key relationships involving measurements and their corresponding attributes.

Relationship	Source Class	Target Attributes
`has_image_value`	`Images_Measurement`	`Is Object Visible`, `Images`, `Real Time Framerate`
`has_spectrum_value`	`Spectrum_Measurement`	`spectrum.X`, `spectrum.Y`, `Spectrum Wavelength`, `Spectrum Width`, `Gabarits_Activated_Global_Sup_Inf`, `Gabarits_Secured_Global_Sup_Inf`
`characterized_by`	`Images_Measurement`	`Energy & Power values`, `Centroid`, `Image.pointing stability values`, `Profiles Widths H and V`

Table 4.7: Measurement Relationships

4.3.3 Derived Relationships:

Derived relationships provide secondary associations that enrich the ontology's semantic model. Table 4.8 provides examples of these relationships.

Relationship	Source Class	Target Attributes
has_stability_value	Image.pointing stability values	Reference Pointing Stability Values, Vertical Pointing Stability Values, Horizontal Pointing Stability Values, Total Pointing Stability Values, Timer OVer Pointing Stability Values
contains	Profiles Widths H and V	HprofileLine, VprofileLine
has_energy_value	Energy & Power	Energy or Power Value, Energy or Power Timer Over, Energy or Power Unit
has_centriod_value	Centroid	Energy.Centroid, Geometric.Centroid
has_Vprofile_value	VprofileLine	VprofileLine.X, VprofileLine.Y
has_Hprofile_value	HprofileLine	HprofileLine.X, HprofileLine.Y

Table 4.8: Derived Relationships

4.4 Integration with Data Storage:

The ontology bridges experimental data and storage systems through specific classes and identifiers:

4.4.1 Dataset Storage:

- Dataset is linked to storage endpoints, such as mongoDB and MinIO, ensuring data accessibility.
- Relationships like values_stored and images_stored map data from measurements to their respective storage repositories.

4.4.2 Data Retrieval:

- Queryable attributes and relationships allow for efficient retrieval of experimental data based on specific parameters or conditions.

4.5 Relationships and Dependencies

4.5.1 Protégé

Protégé was utilized as the primary tool for designing and visualizing the ontology. Its graphical interface and extensibility made it ideal for creating a well-structured ontology tailored to DRACO's needs.

4.5.2 RDFLib

RDFLib, a Python library, was employed to parse and query the ontology. Its compatibility with OWL and RDF formats enabled seamless integration with other tools and the database.

4.5.3 OWL (Web Ontology Language)

The ontology was implemented in OWL, which provided a standardized framework for defining the classes, relationships, and properties within the DRACO experiment.

4.6 Summary

In this chapter, we examined the ontology-driven approach for structuring DRACO's experimental data. By leveraging ontologies, DRACO ensures a standardized, machine-readable representation of relationships between experiments, devices, and measurements.

The implementation of semantic metadata and linked data principles allows for efficient querying and reasoning over datasets, significantly enhancing data accessibility. Additionally, we discussed the role of knowledge graphs in visually representing these relationships.

The next chapter will focus on the development of the DRACO frontend, where these structured datasets are transformed into interactive visualizations to facilitate intuitive data exploration.

5

Frontend Development for DRACO Datamaster

5.1 Motivation

The rapid growth of modern data-driven approaches in scientific research provides a foundation for deriving insights from large datasets; while powerful, it often lacks transparency and interoperability, primarily when the results are derived from machine learning, Data mining, and cluster analysis. The techniques are inefficient in explaining the reasoning behind their results, exposing the "black-box" nature of analytics; this leads to eroding trust in the conclusions being drawn, especially when decisions or hypotheses depend on the outputs of the process [35][37].

5.1.1 Challenges in Interpretability and Transparency

Scientific research is increasingly data-driven, requiring efficient systems to process, analyze, and visualize large volumes of complex experimental Data. The DRACO experiment generates a significant amount of multi-dimensional Data from multiple devices, including cameras, spectrometers, and diagnostics tools. However, extracting meaningful insights from this data poses several challenges:

- **Complexity of Methods**: Advanced analytics often relies on complex models that are difficult for researchers to understand or communicate intuitively [37].

- **Opaque Decision-Making**: Outputs may lack explanatory frameworks, making it hard to justify why a particular result was achieved [14].
- **Data Overload**: As datasets grow, traditional analysis tools struggle to synthesize relationships and patterns in a manner accessible to researchers [56].

Dashboard-based data visualization offers a solution to these issues by providing real-time, structured, and interactive views of complex datasets [48]. Dashboards, when well-designed, enable researchers to quickly identify key patterns while maintaining a high-level overview of experimental data [3]. Studies on dashboard usability emphasize that clear data organization and interactive components significantly reduce cognitive load and improve decision-making efficiency [30].

5.1.2 Interactive Visual Exploration as a Solution

To address these challenges, the DRACO DataMaster frontend integrates interactive dashboards that offer:

- **Real-time filtering and exploration** to dynamically adjust views based on selected parameters.
- **Graph-based relationships** that visually connect experiments, devices, and results.
- **Adaptive visualizations** that allow users to zoom into specific details while maintaining a broader overview.

This approach aligns with the principles of the modern dashboard, emphasizing multiview coordination, contextual awareness, and decision support [48]. Unlike traditional dashboards, which primarily focus on KPI tracking, the DRACO frontend incorporates visual analytics techniques, allowing researchers to interactively explore multi-level scientific data structures [3].

A recent study on dashboard design patterns categorizes effective dashboards into narrative, analytical, and exploratory genres [30].

The DRACO frontend aligns closely with analytical dashboards, enabling scientists to examine data both at a high level and in granular detail. Key design principles applied include:

- Minimizing cognitive load by presenting only the most relevant information [30].
- Supporting multiple levels of analysis through overview-detail mechanisms [3].
- Using structured layouts to facilitate rapid information retrieval [48].

Moreover, best practices in effective dashboard design recommend:

- **Avoiding visual clutter** by implementing progressive disclosure, where detailed information is displayed only upon interaction [30].
- **Enhancing usability** through intuitive data filtering, search, and zooming [3].
- **Utilizing color coding and anomaly detection** to highlight critical data points [48].

In summary, DRACO DataMaster's frontend transforms raw experimental data into an interactive, decision-supporting system, providing researchers with a more efficient, insightful, and structured way to analyze complex scientific experiments.

5.2 Design Goals for the DRACO Frontend

The DRACO frontend system is designed to bridge the gap between complex experimental datasets and actionable insights. Its primary objective is to provide researchers with an intuitive, interactive platform for visualizing and exploring data relationships. By leveraging advanced visualization techniques and interactive tools, the system ensures that researchers can better understand and utilize experimental data in their workflows.

5.2.1 Core Objectives

The DRACO experiment generates high-dimensional experimental data from multiple sources, requiring an intuitive and efficient system for visualization. The primary objective of the DRACO frontend is to provide researchers with an interactive platform that enhances data exploration, interpretability, and decision-making. Modern dashboards in scientific research must meet specific design principles to be effective [48, 42]. The core objectives of the DRACO frontend include:

- **Enhancing Data Interaction:** Users should be able to dynamically filter, zoom, and explore experimental data through intuitive UI elements.
- **Supporting Multi-Level Data Representation:** The interface must balance overview and detail, allowing users to start with a high-level summary and drill down into specific experimental data [49].
- **Ensuring Scalability and Performance:** Given the volume of DRACO data, the system must support efficient data retrieval, caching, and rendering [17].
- **Facilitating Decision-Making:** The frontend should highlight key patterns, anomalies, and trends to help researchers interpret results faster.

By incorporating these principles, the DRACO front-end follows best practices in scientific visualization and human-computer interaction (HCI), ensuring usability and efficiency [18].

5.2.2 Interaction and Usability Principles

Effective data exploration relies on well-established UX and interaction principles. The DRACO frontend applies several human-computer interaction (HCI) laws to improve usability:

- **Hick's Law:** Reducing the number of visible choices speeds up decision-making. The front end initially displays only essential options, with advanced controls available on demand [10].

- **Gestalt Principles:** Elements that are visually grouped appear more related. The DRACO dashboard clusters experiment components using color coding, spacing, and node-link structures [56].
- **Fitts' Law:** Larger, closer UI elements are more straightforward to interact with. Key actions (e.g., selecting experiments, filtering results) are designed with significant, clickable elements for accessibility [38].
- **Shneiderman's Visual Information-Seeking Mantra:** "*Overview first, zoom and filter, then details on demand.*" The DRACO front end follows this approach to ensure effective multi-scale visualization [49].

These principles ensure that the DRACO frontend remains intuitive, reducing user cognitive load while maximizing efficiency.

5.2.3 Visualization Techniques in Scientific Dashboards

Scientific dashboards require specialized visualization techniques beyond standard UI design. DRACO DataMaster implements:

- **Graph-Based Exploration:** The front-end uses knowledge graphs (PyVis) to visually represent experiment relationships. This allows users to trace dependencies between devices, measurements, and runs [7].
- **Time-Series and Parametric Visualization:** Experimental parameters are plotted using dynamic X vs. Y plots selected by the user to analyze trends and anomalies [42].
- **Hierarchical Data Representation:** DRACO's data is inherently hierarchical, requiring multi-level visualizations that maintain contextual awareness across experiments [17].
- **Color Mapping for Scientific Visualization:** The frontend follows Paul Tol's colorblind-safe palette to ensure accessibility in visual analytics [55].

By integrating these advanced techniques, DRACO DataMaster aligns with state-of-the-art scientific visualization principles and dashboard design best practices [18, 3].

5.2.4 Performance Considerations

Given the large-scale data managed by DRACO DataMaster, the front end must be optimized for performance. Several techniques have been implemented to ensure efficiency in data rendering and retrieval:

- **Lazy Loading:** Data is fetched on demand to minimize initial load times, improving responsiveness.
- **Asynchronous Data Fetching:** Queries are executed asynchronously, preventing UI freezing when handling large datasets [12].
- **Client-Side Caching:** Frequently accessed data is stored locally to reduce repeated queries.
- **Graph Optimization:** The knowledge graph visualization uses **force-directed layouts** to prevent node overlap and ensure clarity [3].

These strategies enhance the front end's ability to handle thousands of interconnected data points while maintaining smooth user interactions.

5.3 UX & Visual Design Laws in DRACO Frontend

5.3.1 Foundational UX Principles

The DRACO frontend applies well-established User Experience (UX) and Visual Analytics principles to optimize usability, efficiency, and accessibility in scientific data exploration.

- **Shneiderman's Mantra:** "*Overview first, zoom and filter, then details on demand*" ensures multi-level data exploration [49].
- **Hick's Law:** Reducing the number of choices in the interface improves decision speed and usability [53].
- **Fitts' Law:** Larger, closer UI elements improve usability and interaction speed [38].
- **Gestalt Principles:** Color, grouping, and spacing create **visual hierarchy** and improve readability [57, 45].

- **Pre-attentive Processing:** Color coding and shape differentiation help users recognize data patterns quickly [56].
- **Aesthetic-Usability Effect:** Enhancing visual appeal increases usability, aligning with research showing that users perceive aesthetically pleasing interfaces as easier to use [23].
- **Tesler's Law of Complexity Conservation:** The DRACO frontend reduces cognitive load by handling backend complexity while maintaining a simple UI [43].
- **Doherty Threshold:** System response time remains below `400ms` to keep users engaged, optimizing real-time data retrieval from MongoDB and MinIO [13].
- **Progressive Disclosure:** Advanced controls and settings appear only when necessary, reducing UI clutter [53].

By integrating these UX laws, the DRACO front end ensures intuitive, efficient, and scalable scientific data visualization.

5.3.2 Shneiderman's Visual Information-Seeking Mantra

Shneiderman's Visual Information-Seeking Mantra is implemented in the DRACO frontend to facilitate multi-level data exploration [49]:

- **Overview:** The interactive knowledge graph provides a `high-level view` of experiments, devices, and measurements.
- **Zoom and Filter:** Users can `filter` data based on `Experiment`, `Run`, and `Shot_ID`, dynamically refine their focus.
- **Details on Demand:** Clicking a node retrieves `real-time metadata, measurements and plots` from MongoDB and MinIO.

This approach allows researchers to move seamlessly between broad overviews and in-depth analysis.

5.3.3 Hick's Law: Reducing Cognitive Load

Hick's Law states that *the time required to make a decision increases with the number of choices available* [53]. To optimize usability, the DRACO frontend:

- **Uses a progressive selection process**: Users first choose an experiment, then a run, then a shot ID—avoiding overwhelming them with too many options at once.
- **Simplifies UI controls**: The interface hides non-essential settings until they are needed, reducing distractions.

By structuring interactions in this way, the system remains streamlined and efficient.

5.3.4 Fitts' Law: Optimized Interaction Design

Fitts' Law states that *the time to reach a target is a function of its size and distance* [38]. The DRACO frontend incorporates this principle by:

- Using large, well-spaced buttons for experiment and shot selection.
- Optimizing clickable elements in the knowledge graph, ensuring users can interact efficiently.

These design choices minimize user effort, enhancing workflow efficiency.

5.3.5 Gestalt Principles: Enhancing Visual Hierarchy

Gestalt Principles describe how humans perceive structured patterns rather than isolated elements [57]. DRACO applies these principles by:

- Grouping related UI elements together (e.g., experiment selection, data tables).
- Color coding different data types (e.g., blue for experiments, green for runs, orange for shots).
- Using edge thickness in graphs to indicate strong relationships.

These techniques improve **data readability and user comprehension**.

5.3.6 Tesler's Law: Simplifying Complexity

Tesler's Law states that *every system has an inherent complexity that cannot be removed but should be hidden from the user* [43]. In DRACO:

- Complex backend operations (e.g., MongoDB queries, MinIO image retrieval) are **handled automatically**, keeping the frontend simple.
- Users interact with **a structured UI** instead of manually searching for files or writing database queries.

This ensures that researchers focus on insights rather than system mechanics.

5.3.7 Doherty Threshold: Keeping Users Engaged

The Doherty Threshold states that *system response times should remain below 400ms* to keep users engaged [13]. The DRACO frontend achieves this by:

- **Asynchronous data fetching:** MongoDB and MinIO queries are optimized to load data quickly.
- **Lazy loading of images and tables:** Ensures that only necessary data is retrieved at a time.

These optimizations prevent lag and improve user satisfaction.

5.3.8 Pre-attentive Processing: Rapid Recognition of Key Information

Pre-attentive processing allows users to recognize patterns *before conscious thought* [56]. DRACO applies this through:

- **Color-coded device activity indicators** (e.g., ✓ = active, ✗ = inactive).
- **Shape differentiation** for experiments, runs and shot nodes in the knowledge graph.

- **Size scaling** to emphasize key data points.

This enhances user efficiency and data interpretation speed.

5.3.9 Progressive Disclosure: Reducing UI Clutter

Progressive Disclosure ensures that *advanced settings appear only when necessary,* preventing cognitive overload [53]. In DRACO:

- Users first interact with basic controls (experiment selection, graph view).
- Advanced options (e.g., timestamp filtering, image annotations) appear dynamically based on user selections.

This keeps the interface clean and user-friendly.

5.3.10 Conclusion

The UX and visual design principles applied in DRACO DataMaster significantly enhance usability, accessibility, and data interpretability. By incorporating Shneiderman's Mantra, Hick's Law, Fitts' Law, Tesler's Law, the Doherty Threshold, and Progressive Disclosure, the system ensures:

- Efficient data exploration through structured interaction flows.
- Improved usability by balancing simplicity with advanced functionality.
- Scalability for complex datasets, supporting high-energy physics research.

5.4 Visual Analytics Techniques in DRACO Frontend

5.4.1 Introduction to Visual Analytics

Visual analytics (VA) combines interactive data visualization with computational data analysis to enhance human decision-making. It

provides a way to process, explore, and extract insights from complex datasets by leveraging machine learning techniques and intuitive human-computer interactions [35]. The DRACO front end integrates VA techniques to facilitate the exploration of high-energy physics experiments, ensuring researchers can efficiently interpret large-scale experimental data.

5.4.2 Knowledge Graphs for Data Exploration

Knowledge graphs are crucial in visualizing relationships between experimental components in DRACO. They allow researchers to explore hierarchical data structures, revealing interdependencies among experiments, devices, and measurements.

Graph visualization problem: Large-scale networks often suffer from node overlap and clutter, making it challenging to interpret relationships. To address this, the DRACO frontend employs force-directed layouts, an established approach for network visualization [36, 6].

What is a Force-Directed Layout?

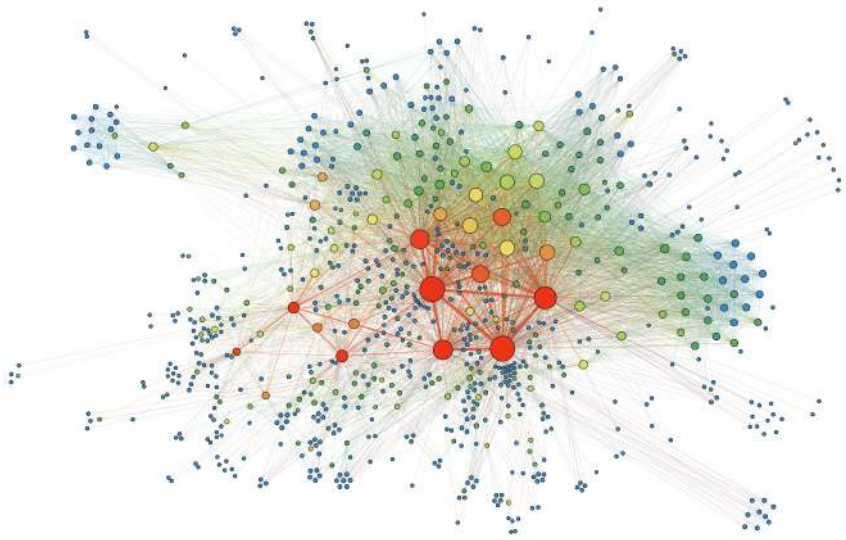

Figure 5.1: Force-Directed Layout Applied to Knowledge Graph Visualization, adapted from [20].

Force-directed layouts use **physics-based simulations** to dynamically arrange nodes. Instead of fixed placements, this approach applies attractive and repulsive forces to ensure an optimal structure [28].

- **Nodes act as charged particles**: They repel each other to minimize clutter.
- **Edges act as springs**: They pull related nodes closer to reflect relationships.
- **Self-adjusting Layout**: The visualization adapts dynamically to new data.

Why Use Barnes-Hut Force-Directed Placement?

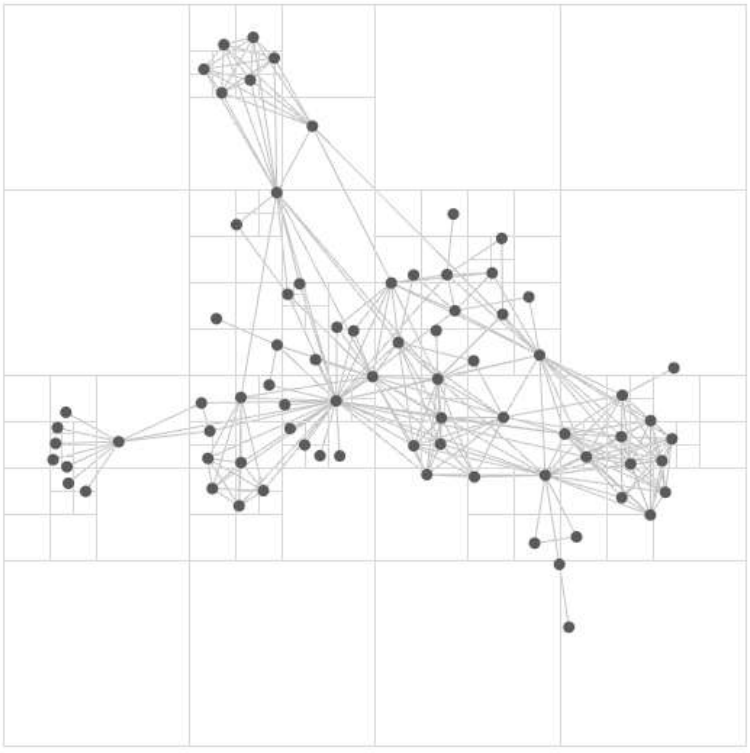

Figure 5.2: Barnes-Hut Quadtree Approximation for Force-Directed Layouts, adapted from [24].

Traditional force-directed methods can be computationally expensive ($\mathcal{O}(N^2)$). DRACO implements the **Barnes-Hut Approximation**,

which reduces complexity to $\mathcal{O}(N \log N)$, making real-time visualization feasible [4].

Key Benefits:

- **Scalability**: Efficient for thousands of nodes.
- **Clarity**: Reduces overlap, improving readability.
- **Hierarchical Structuring**: Enhances interpretation of nested relationships.

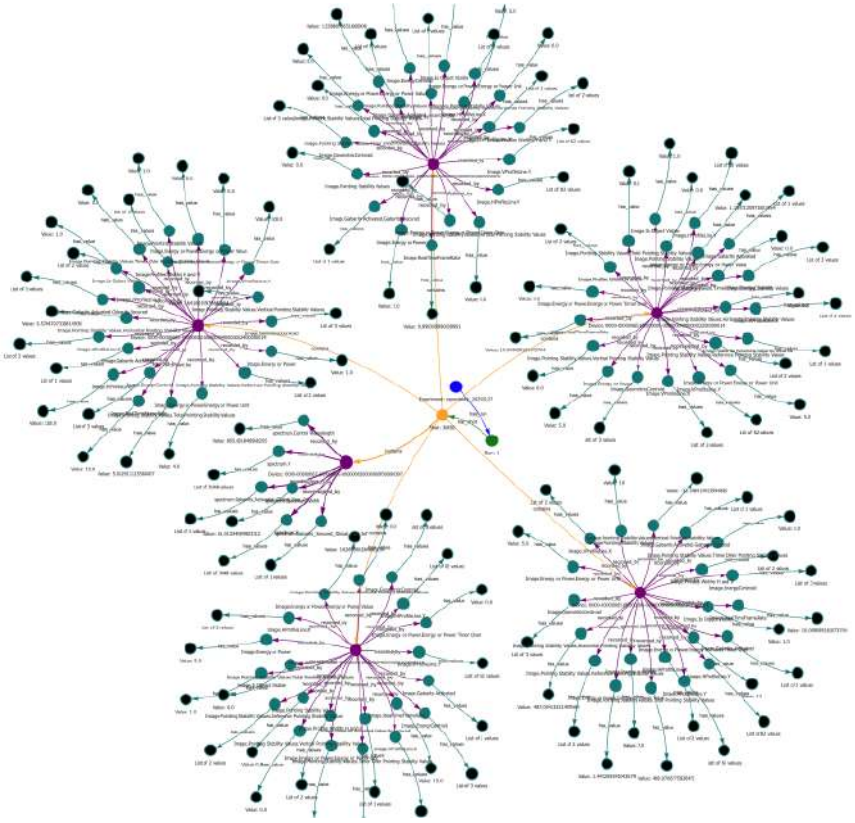

Figure 5.3: Knowledge Graph Representation in DRACO Frontend

Implementation in DRACO Datamaster

The **PyVis** library applies Barnes-Hut force-directed placement to visualize experiment-data-device relationships.

- **Clear Separation**: Related nodes stay clustered while unrelated nodes remain spaced apart.

- **Real-time Adjustments**: Users can dynamically zoom, pan, and filter data.
- **Hierarchical Organization**: Displays relationships in an intuitive manner.

This structured approach significantly enhances the usability of DRACO's knowledge graph, aligning with modern best practices in network visualization [36, 28]. Researchers can explore complex relationships more intuitively by reducing clutter and optimizing interactions.

5.4.3 Parameterized Measurement Plots

The DRACO frontend supports **dynamic parameterized measurement plots** that enable researchers to analyze correlations between experimental parameters. Users can:

- Select custom X and Y parameters for visualization.
- Apply filtering and zooming to refine data exploration.
- Identify outliers, trends, and anomalies.

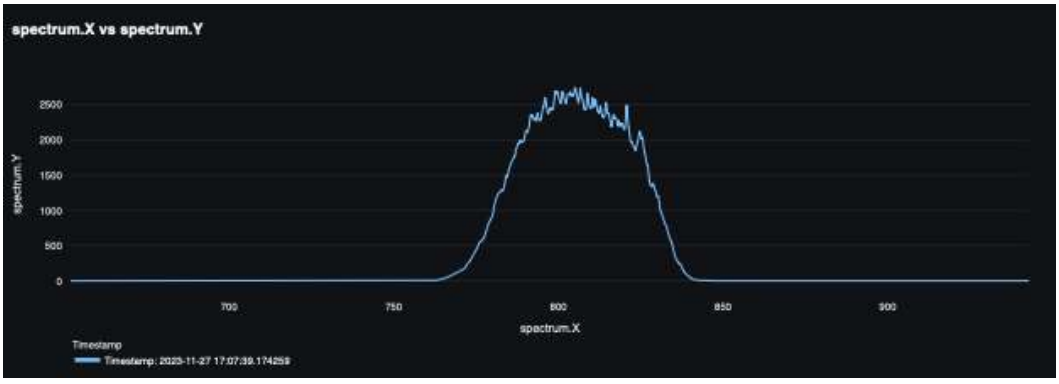

Figure 5.4: Example of Parameterized Measurement Plot

These visualizations are crucial for experimental research as they help in:

- **Hypothesis Validation:** Scientists can confirm experimental assumptions by analyzing parameter relationships.

- **Anomaly Detection:** Unexpected data points can be easily identified and further investigated [14].
- **Interactive Exploration:** The system implements **Hick's Law** by simplifying parameter selection, reducing cognitive load [53].

5.4.4 Device Activity Tables for Operational Status Tracking

To complement the graphical visualization techniques, the DRACO frontend includes a **device activity table** that provides an overview of operational device statuses. This structured format offers:

- **Binary activity indicators:** Green ✓ for active devices, Red ✗ for inactive devices.
- **Time-based tracking:** Users can track device performance over multiple shots.
- **Quick filtering options:** The UI follows Progressive Disclosure, revealing additional insights as users interact with specific devices [43].

Figure 5.5: Device Activity Table for Status Monitoring

The combination of structured tables and interactive graphics ensures a multi-perspective analysis of experimental data.

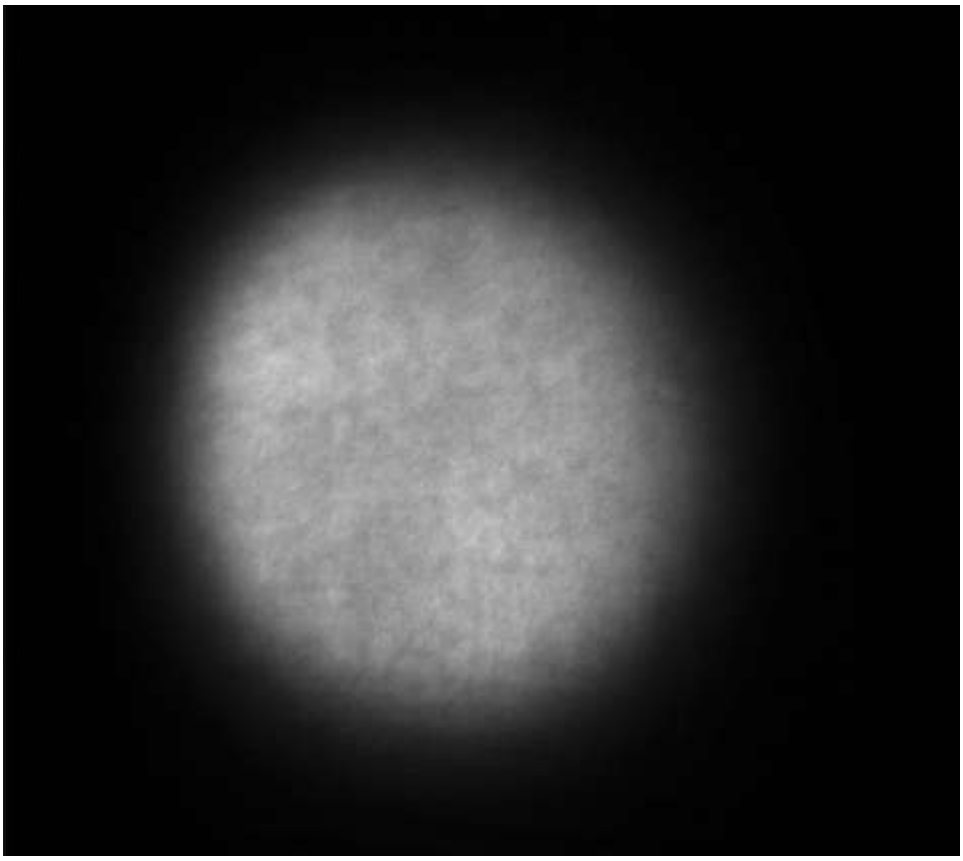

Figure 5.6: Image-Based Analysis in DRACO Frontend

5.4.5 Image-Based Data Interpretation

Beyond numerical data visualization, the DRACO frontend enables researchers to retrieve and analyze experimental images stored in MinIO. This feature allows users to:

- Compare image-based observations with numerical data.
- Identify experimental conditions that may influence measurement results.
- Enhance collaborative research by integrating visual evidence into discussions.

The implementation of image retrieval aligns with **Tesler's Law**, abstracting complexity by automating the fetching process rather than requiring manual search operations [43].

5.4.6 Color Theory and Accessibility in Visualization

Ensuring accessibility in visualization is crucial for usability across diverse research teams. Studies indicate that approximately **8% of men** and **0.5% of women** have some form of **color vision deficiency** (CVD) [32]. Traditional red-green color schemes often create confusion for individuals with **deuteranopia**, the most common form of color blindness.

Using Paul Tol's Muted Color Palette

To address these concerns, the DRACO frontend implements a **colorblind-friendly palette** developed by Paul Tol [55]. This scheme is optimized for:

- High perceptual distinctiveness, ensuring clear differentiation between elements.
- Avoiding problematic color pairs, such as red-green, which are difficult to distinguish for deuteranopia.
- Maintaining aesthetic quality means making the visualization intuitive and visually appealing.

Mapping Colors to DRACO Visualization Elements

To ensure optimal readability, the following mappings have been applied as presented in 5.1:

High-Contrast Themes for enhanced readability, following best practices in data visualization [31].

These enhancements improve data interpretability and inclusivity.

5.4.7 Conclusion

The DRACO front end leverages cutting-edge visual analytics techniques to enhance data exploration in high-energy physics research. By integrating:

- Knowledge graphs for contextual navigation.

Element	Color Used	Reasoning
Experiments	Blue	Associated with trust and stability [56].
Runs	Green	Represents activity and progression [9].
Shot Counters	Orange	Ensures clear separation as hierarchical connectors [5].
Devices	Purple	Distinctive and perceptually distant from other colors.
Measurements	Teal	Enhances visibility while avoiding conflicts with red/green [31].
Values	Dark Red	Emphasizes critical insights without conflicting with accessibility guidelines.

Table 5.1: Color Mapping Based on Paul Tol's Muted Palette

- Parameterized plots for trend analysis.
- Device activity tables for operational tracking.
- Image-based analysis for experimental validation.
- Colorblind-friendly design for accessibility.

The system ensures an interactive, efficient, and scalable approach to scientific data interpretation.

5.5 Practical Example: A Full Workflow

To illustrate how the DRACO front end supports researchers in scientific data exploration, this section provides a step-by-step example demonstrating its key functionalities. As illustrated in Figure 5.7, the workflow follows a typical scenario where a scientist investigates an unexpected anomaly in experimental data.

The major workflow components include:

- **Data Retrieval**: Experimental data is queried from MongoDB and image data is fetched from MinIO.

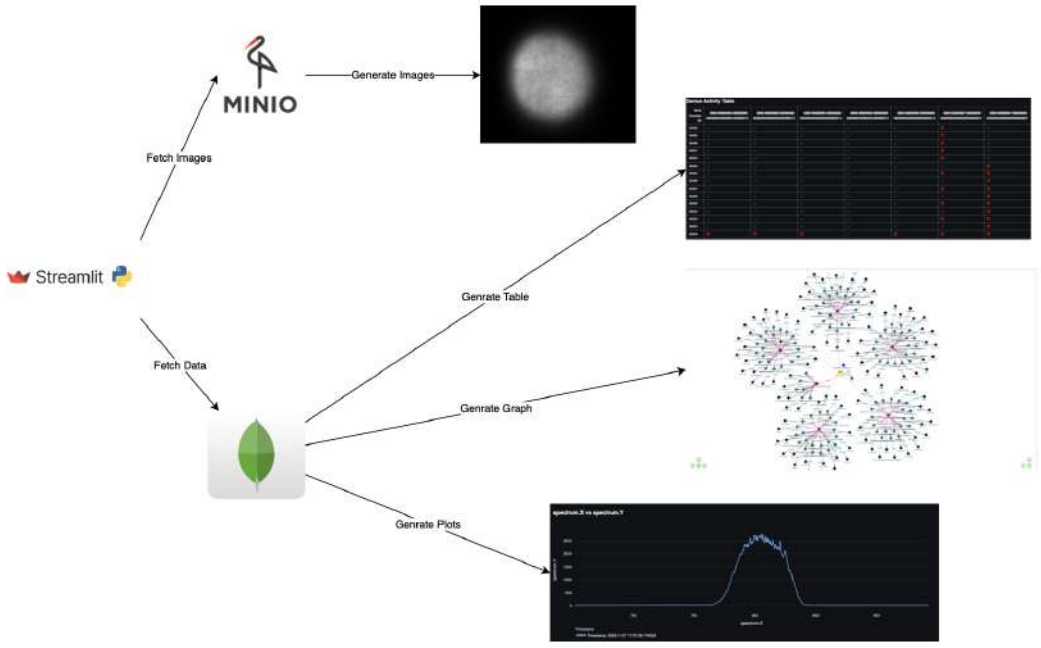

Figure 5.7: Interactive Knowledge Graph in DRACO Frontend

- **Graph Generation**: The knowledge graph is dynamically built using PyVis, mapping relationships between experiments, devices, and measurements.
- **Image Visualization**: The front integrates image processing capabilities to display experimental images retrieved from **MinIO**, enhancing visual data analysis.
- **Tabular Representation**: Device activity tables summarize active/inactive devices.
- **Parameterized Measurement Plots**: Users can select X and Y-axis parameters dynamically for correlation analysis.
- **User Interactivity**: The UI updates dynamically using Streamlit widgets.

To achieve these goals, the following technologies are used:

Category	Technology	Purpose
Frontend Framework	Streamlit	Provides an interactive web-based UI with minimal setup [51].
Graph Visualization	PyVis (Vis.js)	Creates interactive knowledge graphs for data exploration [22].
Plotting Library	Plotly	Enables interactive measurement plots (X vs. Y correlation analysis) [42].
Database	MongoDB	Stores experimental data in a **flexible JSON-based structure**.
Object Storage	MinIO	Manages and retrieves large image datasets efficiently.
Data Handling	Pandas	Processes tabular data for device activity tables [41].

Table 5.2: Technologies Used in DRACO Frontend

5.5.1 Scenario: Identifying an Experimental Anomaly

Consider a scientist analyzing an unexpected **energy spike** in a DRACO experiment. The goal is to investigate the anomaly by exploring relationships between devices, runs, and measured parameters.

The workflow follows Shneiderman's **Visual Information-Seeking Mantra**—*"Overview first, zoom and filter, then details on demand"* [49]—ensuring efficient data navigation.

Step 1: Selecting an Experiment

Upon opening the DRACO front end, the researcher is presented with an **Select Experiment** option, which presents details of all the previous experiments `Experiment_ID` carried out using the DRACO Experimental setup. As illustrated in Figure 5.8, the experiments are labeled by the date they are carried out, giving the researchers an option to view the experimental data of a particular date. This allows the user to:

- Get an overview of all recorded experiments.
- Quickly navigate to the experiment of interest by clicking on a node.

Figure 5.8: Select Experiment ID in DRACO Frontend

Step 2: Filtering Data for a Specific Run

After selecting an experiment, the researcher can narrow down the search by selecting:

- A specific `Run_ID` (e.g., "Run_1").
- Every experiment consists of at least one run.

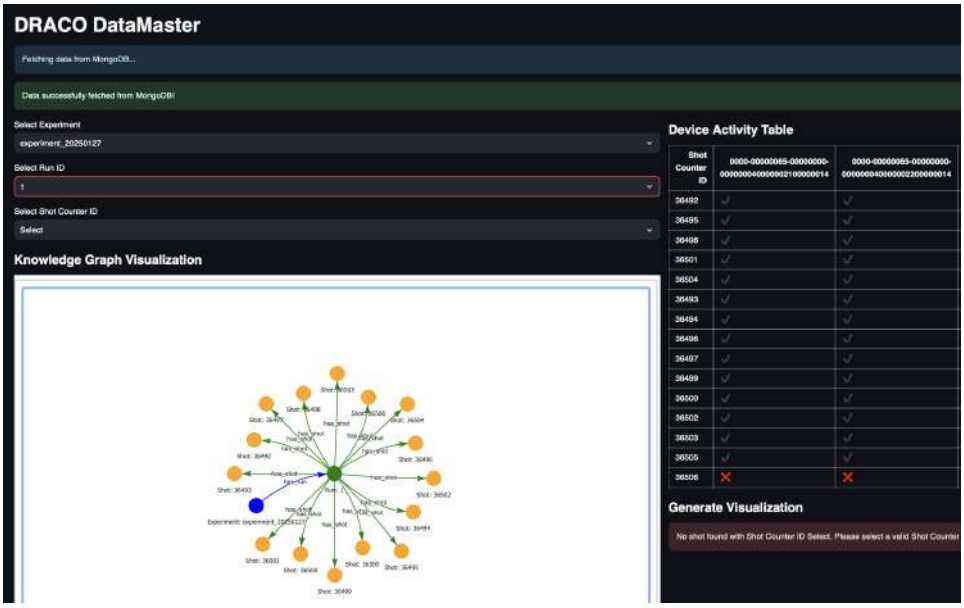

Figure 5.9: Device Activity Table in DRACO Frontend

As illustrated in Figure 5.9 UI dynamically updates, displaying:

- The **device activity table** (highlighting active/inactive devices).

- Intial Knowledge Graph consisting of `Run_ID`, `Experiment_ID` and `Shot Counter ID` related to them.

This implementation aligns with **Hick's Law**, reducing cognitive load by presenting only relevant options at each stage [53].

Step 3: Inspecting Device Activity and Status

The next step is to check if the anomaly is related to a device malfunction. The **device activity table** provides a structured view of:

- The operational status of all devices used in the `experiment`.
- A timeline of active/inactive devices across multiple shots.

Figure 5.10: Device Activity Table in DRACO Frontend

The table follows Pre-attentive Processing principles by using:

- ✓ is an indicator for active devices.
- ✗ is an indicator for inactive devices.

This allows researchers to recognize device malfunctions instantly, reducing time spent on manual checks [56].

Step 3.5: Selecting the Appropriate Shot Counter ID

Once the device activity table is reviewed, the researcher must select an appropriate `Shot Counter ID` based on device availability. This step ensures that:

- The researcher first reviews the **Device Activity Table** to identify active/inactive devices.
- Based on this, an appropriate `Shot Counter ID` is determined.
- The researcher then selects the following `Shot Counter ID` from the dropdown menu.

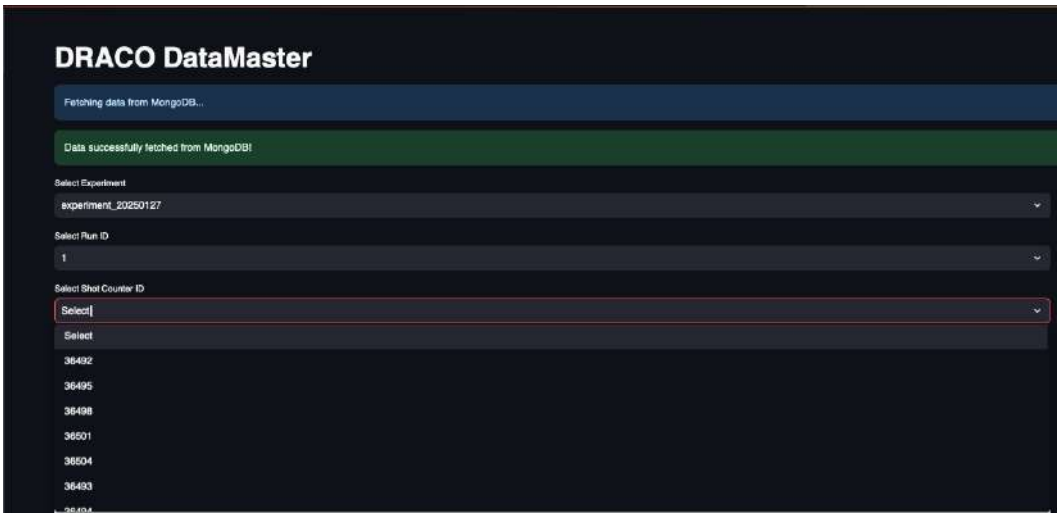

Figure 5.11: Select Experiment ID in DRACO Frontend

The DRACO frontend dynamically updates the dropdown list to reflect available shot counters, as shown in Figure 5.11.

Step 4: Generating and Interpreting the Knowledge Graph

After selecting the appropriate `Shot Counter ID`, the DRACO frontend dynamically generates a **Knowledge Graph** visualization. This graph shows the relationships between experimental components, including devices, measurements, and runs.

The Knowledge Graph provides the following insights:

- **Device Connections**: Visualize how different devices are linked to measurements and experiments.
- **Measurement Relationships**: Understand which measurements are associated with specific devices and runs.

- **Data Flow**: Trace the flow of data through various components of the experiment.

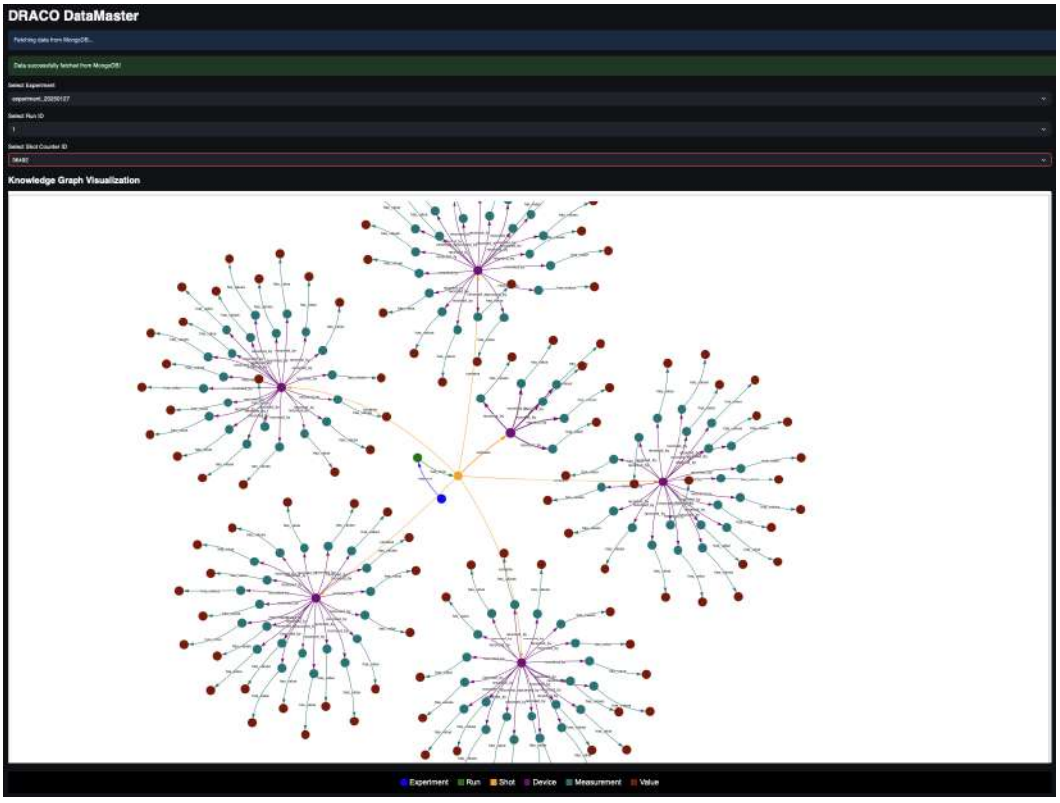

Figure 5.12: Knowledge Graph Visualization of DRACO Experiment

As shown in Figure 5.12, the graph uses a node-link structure to represent entities and their relationships. The color coding helps distinguish between different types of nodes:

- **Blue**: Experiment
- **Green**: Run
- **Orange**: Shot Counter
- **Purple**: Device
- **Teal**: Measurement
- **Red**: Value

This visual representation aids researchers in identifying complex dependencies and patterns in the experimental data. It aligns with

Shneiderman's Mantra—providing an overview, allowing for zooming into details, and enabling on-demand information [49].

Once the Knowledge Graph is reviewed, researchers can analyze specific experimental measurements in the next step.

Step 5: Analyzing Experimental Measurements

To explore the anomaly further, the researchers must select two parameters to plot. First, we need to select the device from the dropdown, which will then dynamically update the X and Y axis measurement dropdown with the corresponding measurements related to that device. In the Figure 5.13 the device chosen is a spectrometer:

- **X-axis:** `Spectrum.X` is being choosen.
- **Y-axis:** `Spectrum.Y` is being chossen.

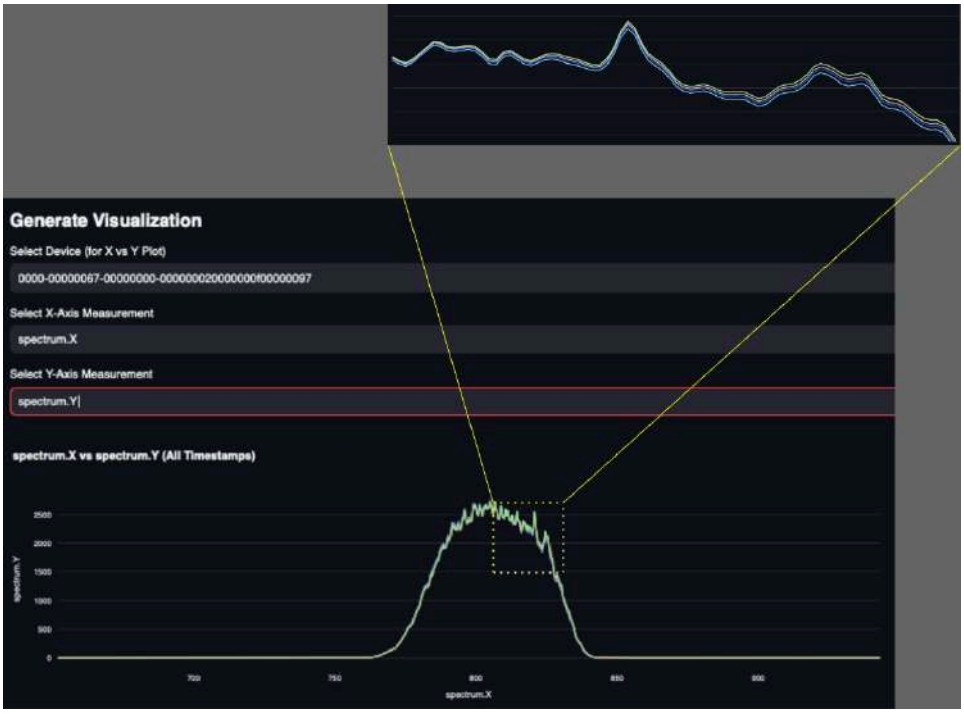

Figure 5.13: Parameterized Measurement Plot in DRACO Frontend

A dynamic scatter plot is generated (Figure 5.13), enabling researchers to:

- Apply zooming and filtering to focus on specific data points as shown in Figure 5.13.
- It helps to identify trends, correlations, or outliers.
- As shown in Figure 5.13, we can use the play and pause option to check specific anomalies.

The plot supports **Fitts' Law** by ensuring that all interactions are optimized for ease of selection [38].

Step 6: Retrieving and Analyzing Experimental Images

The researchers can retrieve experimental images stored in **MinIO** to gain additional insights. The system:

- Displays images corresponding to the selected shot.
- Allows side-by-side comparisons with the knowledge graph generated based on the experiment runs.
- Supports zooming and annotations for detailed analysis.

This step aligns with **Tesler's Law**, abstracting complexity by automatically fetching images instead of requiring manual file searches [43].

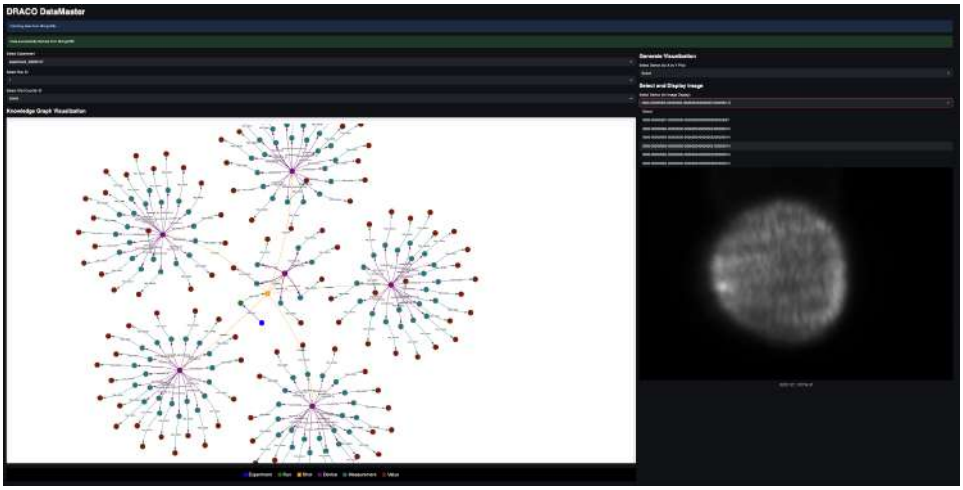

Figure 5.14: Experimental Image Retrieval in DRACO Frontend

Step 7: Documenting and Exporting Insights

After completing the analysis, the scientist can:

- Export plots and tables as reports for further review.
- Save filtered data views for later comparison.
- Share insights with collaborators.

The interface is designed with the Aesthetic-Usability Effect in mind, ensuring that visually appealing layouts improve usability [23].

5.5.2 Conclusion

This example illustrates how the DRACO front end supports an efficient, structured workflow for scientific data analysis. By integrating:

- **Shneiderman's Mantra** for multi-level exploration.
- **Hick's Law** for simplified navigation.
- **Fitts' Law** for optimized interactivity.
- **Gestalt Principles** for clear data representation.
- **Tesler's Law** for reducing system complexity.
- **Pre-attentive Processing** for rapid recognition.

The system enhances the interpretability of experimental data, making it easier for researchers to detect anomalies, analyze trends, and make informed decisions.

5.6 Conclusion

5.6.1 Key Contributions

The following contributions highlight the effectiveness of the DRACO frontend:

- **Intuitive Data Navigation:** The application of Shneiderman's Mantra ensures seamless transitions from high-level overviews to detailed analysis [49].

- **Optimized Usability:** By incorporating Hick's Law, Fitts' Law, and Progressive Disclosure, the system minimizes cognitive load and interaction effort [53, 38].
- **Enhanced Visualization:** The knowledge graph, device activity tables, and measurement plots use Gestalt Principles and Preattentive Processing to improve data readability [56, 57].
- **Scalability and Performance:** Asynchronous data fetching lazy loading, and optimized query handling ensure fast response times, aligning with the Doherty Threshold [13].
- **AI-Driven Insights:** Future enhancements will integrate machine learning for anomaly detection and predictive analytics, reducing the manual effort needed for data interpretation [14].
- **Collaborative Science:** Planned multi-user features, such as real-time annotations and cloud-based experiment sharing, align with the FAIR data principles [58].

5.6.2 Lessons Learned

During development, several key lessons were identified:

- **Balancing Simplicity and Complexity: Tesler's Law** was critical in ensuring that complex backend operations (e.g., MongoDB queries, MinIO storage retrieval) were abstracted from the user while maintaining flexibility for advanced users [43].
- **Importance of Accessibility:** Designing a colorblind-friendly interface using **Paul Tol's muted color scheme** significantly improved usability [55].
- **Performance Matters:** Efficient data retrieval methods (e.g., caching, parallel queries) play a significant role in maintaining interactivity, especially with large experimental datasets.

5.6.3 Software Dependencies

To ensure the reproducibility of this research, the following software dependencies were used in the development of the DRACO Datamaster system.

Programming Language and Development Environment

- **Python**: Version 3.9.12
- **MongoDB**: Version 5.0.30
- **MinIO**: Version 7.0.0
- **Protégé (Ontology Editor)**: Version 5.5.0 3

Python Libraries

The Python environment used in this project consists of the following libraries and dependencies:

Table 5.3: Software Dependencies and Versions

Library	**Version**
Streamlit	1.42.0
PyVis	0.3.2
Plotly	5.24.1
Pandas	2.2.3
NumPy	2.2.1
SciPy	1.15.1
NetworkX	3.4.2
Matplotlib	3.10.0
MinIO SDK	7.2.15
PyMongo	4.10.1
RDFlib (Ontology Handling)	7.1.1

These dependencies were installed using `pip` and managed within a virtual environment to maintain consistency across different execution platforms.

5.6.4 Final Remarks

Scientific research increasingly depends on interactive, visual, and intelligent data exploration. The DRACO frontend demonstrates that a well-designed user interface, grounded in UX principles and modern visualization techniques, can significantly enhance scientific workflows. By bridging the gap between complex experimental data

and human intuition, DRACO enables researchers to make more informed, data-driven discoveries.

6 Conclusion and Future Work

6.1 Conclusion

The DRACO Datamaster project aims to enhance the visualization, interpretation, and accessibility of complex experimental datasets through the integration of ontology-driven data representation and interactive visual analytics. This thesis introduced a novel approach to managing large-scale experimental data using a knowledge graph visualization system, interactive dashboards, and ontology-based metadata structuring.

To summarize, the key contributions of this work include:

- **Knowledge Graph-Based Data Exploration:** A force-directed layout was used to dynamically structure complex relationships between experimental entities, improving interpretability and reducing data clutter.
- **Interactive Visualization for Anomaly Detection and Validation:** The system enables real-time filtering and parameterized measurements to identify inconsistencies and validate results.
- **Ontology-Driven Data Management:** The structured representation of experimental data ensures interoperability, consistency, and efficient retrieval.

6.1.1 Addressing Research Questions

This thesis was guided by three key research questions:

RQ1: How can knowledge graphs enhance the interpretability of complex experimental datasets?

This work demonstrates that force-directed layouts, such as the Barnes-Hut approximation, significantly improve the visualization of relationships in experimental data. Unlike traditional tabular representations, knowledge graphs dynamically group related entities while ensuring clear separation, making it easier to understand data structures (see Figure 5.3).

RQ2: What role does interactive visualization play in anomaly detection and validation?

The interactive exploration features, including zooming, filtering, and dynamic selection of X-Y measurement plots, enable researchers to spot anomalies and validate results efficiently. Figure 5.13 illustrates how real-time adjustments allow users to compare different datasets and detect outliers in the experimental data.

RQ3: How can an ontology-based framework improve data integration, retrieval, and interoperability?

The ontology-driven approach ensures semantic consistency and structured querying of large-scale experimental data. Through hierarchical entity relationships (see Figure 4.1), users can retrieve specific data points without ambiguity, thereby reducing data redundancy and improving accessibility.

6.2 Future Work

While the current implementation of DRACO Datamaster has successfully demonstrated the feasibility of ontology-driven visualization, several areas can be explored further to enhance functionality, usability, and scalability.

6.2.1 Scalability and Performance Optimization

- **Optimizing Graph Computation:** Implementing further improvements in the force-directed layout algorithm to handle even larger datasets more efficiently.
- **Parallel Processing for Real-Time Updates:** Enabling concurrent computations in backend processing to improve visualization speed.

6.2.2 AI-Assisted Insights and Anomaly Detection

- **Machine Learning for Pattern Recognition:** Integrating AI-based models to detect correlations and anomalies in experimental measurements.
- **Automated Data Quality Validation:** Using unsupervised learning techniques to identify inconsistencies in large datasets.

6.2.3 Improved User Experience and Accessibility

- **Dark Mode and Customizable Themes:** Adding more UI customization options to accommodate user preferences and accessibility needs.
- **Desktop Application Development:** Implementing a cross-platform desktop version of the DRACO Datamaster tool to enable offline data analysis.

6.2.4 Enhanced Ontology Integration and Interoperability

- **Extending Ontology Models:** Expanding the existing ontology framework to support additional experimental parameters and metadata schemas.
- **Integration with External Data Sources:** Allowing DRACO Datamaster to fetch and process data from other research facilities via linked-data principles.

6.3 Final Remarks

This thesis contributes to the field of experimental data management and visualization by integrating knowledge graphs, interactive visual analytics, and ontology-driven frameworks. The results demonstrate that a structured and interactive approach can significantly enhance the interpretability, validation, and accessibility of large-scale experimental datasets.

Future developments, particularly in AI-driven data insights and enhanced interoperability, will further improve the impact of DRACO Datamaster as a robust tool for scientific research.

A

List of Figures

B

List of Tables

C

Bibliography

[1] [ALBERT und THOMAS 2016] F. Albert und A. G. Thomas. **Applications of laser wakefield accelerator-based light sources**. Plasma Physics and Controlled Fusion, Vol. 58(10):103001, 2016.

[2] [ASHBURNER et al. 2000] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig et al. **Gene ontology: tool for the unification of biology**. Nature genetics, Vol. 25(1):25–29, 2000.

[3] [BACH et al. 2022] B. Bach, E. Freeman, A. Abdul-Rahman, C. Turkay, S. Khan, Y. Fan und M. Chen. **Dashboard design patterns**. IEEE Transactions on Visualization and Computer Graphics, Vol. 29(1):342–352, 2022.

[4] [BARNES und HUT 1986] J. Barnes und P. Hut. **A hierarchical O (N log N) force-calculation algorithm**. nature, Vol. 324(6096):446–449, 1986.

[5] [BARTRAM et al. 2017] L. Bartram, A. Patra und M. Stone. **Affective color in visualization**. In: Proceedings of the 2017 CHI conference on human factors in computing systems, 2017, pp. 1364–1374.

[6] [BATTISTA et al. 1998] G. D. Battista, P. Eades, R. Tamassia und I. G. Tollis. **Graph drawing: algorithms for the visualization of graphs**. Prentice Hall PTR, 1998.

[7] [BIZER et al. 2023] C. Bizer, T. Heath und T. Berners-Lee. **Linked data-the story so far**. In: Linking the World's Information: Essays on Tim Berners-Lee's Invention of the World Wide Web, 2023. pp. 115–143.

[8] [BRENNER et al. 2015] C. Brenner, S. Mirfayzi, D. Rusby, C. Armstrong, A. Alejo, L. Wilson, R. Clarke, H. Ahmed, N. Butler, D. Haddock et al. **Laser-driven x-ray and neutron source development for industrial applications of plasma accelerators**. Plasma Physics and Controlled Fusion, Vol. 58(1):014039, 2015.

[9] [BREWER 1994] C. A. Brewer. **Guidelines for use of the perceptual dimensions of color for mapping and visualization**. In: Color hard copy and graphic arts III, Vol. 2171, pp. 54–63. 1994, SPIE.

[10] [COCKBURN et al. 2007] A. Cockburn, C. Gutwin und S. Greenberg. **A predictive model of menu performance**. In: Proceedings of the SIGCHI conference on Human factors in computing systems, 2007, pp. 627–636.

[11] [DANSON et al. 2015] C. Danson, D. Hillier, N. Hopps und D. Neely. **Petawatt class lasers worldwide**. High power laser science and engineering, Vol. 3:e3, 2015.

[12] [DEAN und GHEMAWAT 2008] J. Dean und S. Ghemawat. **MapReduce: simplified data processing on large clusters**. Communications of the ACM, Vol. 51(1):107–113, 2008.

[13] [DOHERTY und THADHANI 1982] W. J. Doherty und A. J. Thadhani. **The economic value of rapid response time**. IBM Report, 1982.

[14] [DOSHI-VELEZ und KIM 2017] F. Doshi-Velez und B. Kim. **Towards a rigorous science of interpretable machine learning**. arXiv preprint arXiv:1702.08608, 2017.

[15] [DRACO FACILITY] **Draco PW laser**. `https://www.hzdr.de/db/Cms?pOid=40859&pNid=2096&pLang=en`. Accessed: 2024-12-12.

[16] [ESAREY et al. 2009] E. Esarey, C. B. Schroeder und W. P. Leemans. **Physics of laser-driven plasma-based electron accelerators**. Reviews of modern physics, Vol. 81(3):1229–1285, 2009.

[17] [FEKETE 2008] J. Fekete. **The value of information visualization**. Information Visualization/Springer, 2008.

[18] [FEW] **Information dashboard design: Displaying data for at-a-glance monitoring**.

[19] [FRUCHTERMAN und REINGOLD 1991] T. M. Fruchterman und E. M. Reingold. **Graph drawing by force-directed placement**. Software: Practice and experience, Vol. 21(11):1129–1164, 1991.

[20] [GRANDJEAN 2015] M. Grandjean. **Introduction à la visualisation de données: l'analyse de réseau en histoire**. Geschichte und Informatik, (18/19):109–128, 2015.

[21] [GRUBER 1995] T. R. Gruber. **Toward principles for the design of ontologies used for knowledge sharing?** International journal of human-computer studies, Vol. 43(5-6):907–928, 1995.

[22] [HAMILTON et al. 2017] W. L. Hamilton, R. Ying und J. Leskovec. **Representation learning on graphs: Methods and applications**. arXiv preprint arXiv:1709.05584, 2017.

[23] [HASSENZAHL 2004] M. Hassenzahl. **The interplay of beauty, goodness, and usability in interactive products**. Human–Computer Interaction, Vol. 19(4):319–349, 2004.

[24] [HEER] **Barnes-Hut Approximation for Force-Directed Layouts**. Accessed: February 18, 2025.

[25] [HEER und SHNEIDERMAN 2012] J. Heer und B. Shneiderman. **Interactive dynamics for visual analysis: A taxonomy of tools that support the fluent and flexible use of visualizations**. Queue, Vol. 10(2):30–55, 2012.

[26] [HOOKER 2013] S. M. Hooker. **Developments in laser-driven plasma accelerators**. Nature Photonics, Vol. 7(10):775–782, 2013.

[27] [HORRIDGE et al. 2009] M. Horridge, S. Jupp, G. Moulton, A. Rector, R. Stevens und C. Wroe. **A practical guide to building owl ontologies using protégé 4 and co-ode tools edition1. 2**. The university of Manchester, Vol. 107, 2009.

[28] [HU 2005] Y. Hu. **Efficient, high-quality force-directed graph drawing**. Mathematica journal, Vol. 12(4):383–408, 2005.

[29] [JACOMY et al. 2014] M. Jacomy, T. Venturini, S. Heymann und M. B. ForceAtlas. **a continuous graph layout algorithm for handy network visualization designed for the Gephi software., 9, e98679**. DOI: https://doi. org/10.1371/journal. pone, Vol. 98679, 2014.

[30] [JANES et al. 2013] A. Janes, A. Sillitti, G. Succi et al. **Effective dashboard design**. Cutter IT journal, Vol. 26(1):17–24, 2013.

[31] [JE und CT 2021] O. JE und T. CT. **Fixing figures for colour blindness**. Nature, Vol. 598:24, 2021.

[32] [JENNY und KELSO 2007] B. Jenny und N. V. Kelso. **Color design for the color vision impaired**. Cartographic perspectives, (58):61–67, 2007.

[33] [JOSHI und KATSOULEAS 2003] C. Joshi und T. Katsouleas. **Plasma accelerators at the energy frontier and on tabletops**. Physics Today, Vol. 56(6):47–53, 2003.

[34] [JOSHI et al. 1981] C. Joshi, T. Tajima, J. Dawson, H. Baldis und N. Ebrahim. **Forward Raman instability and electron acceleration**. Physical Review Letters, Vol. 47(18):1285, 1981.

[35] [KEIM et al. 2008] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas und H. Ziegler. **Visual analytics: Scope and challenges**. Springer, 2008.

[36] [KOBOUROV 2013] S. Kobourov. **Force-directed algorithms**. Handbook of Graph Drawing and Visualization, 2013.

[37] [LIPTON 2018] Z. C. Lipton. **The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.** Queue, Vol. 16(3):31–57, 2018.

[38] [MACKENZIE 1992] I. S. MacKenzie. **Fitts' law as a research and design tool in human-computer interaction**. Human-computer interaction, Vol. 7(1):91–139, 1992.

[39] [MALKA et al. 2008] V. Malka, J. Faure, Y. A. Gauduel, E. Lefebvre, A. Rousse und K. T. Phuoc. **Principles and applications of compact laser–plasma accelerators**. Nature physics, Vol. 4(6):447–453, 2008.

[40] [MANTI et al. 2017] L. Manti, F. M. Perozziello, M. Borghesi, G. Candiano, P. Chaudhary, G. Cirrone, D. Doria, D. Gwynne, R. Leanza, K. Prise et al. **The radiobiology of laser-driven particle beams: focus on sub-lethal responses of normal human cells**. Journal of Instrumentation, Vol. 12(03):C03084, 2017.

[41] [MCKINNEY et al. 2011] W. McKinney et al. **pandas: a foundational Python library for data analysis and statistics**. Python for high performance and scientific computing, Vol. 14(9):1–9, 2011.

[42] [MUNZNER 2014] T. Munzner. **Visualization analysis and design**. CRC press, 2014.

[43] [NORMAN 2013] D. Norman. **The design of everyday things: Revised and expanded edition**. Basic books, 2013.

[44] [NOY et al.] **Ontology development 101: A guide to creating your first ontology**.

[45] [PALMER 1999] S. E. Palmer. **Vision science: Photons to phenomenology**. MIT press, 1999.

[46] [PASSONI et al. 2019] M. Passoni, F. Arioli, L. Cialfi, D. Dellasega, L. Fedeli, A. Formenti, A. C. Giovannelli, A. Maffini, F. Mirani, A. Pazzaglia et al. **Advanced laser-driven ion sources and their applications in materials and nuclear science**. Plasma Physics and Controlled Fusion, Vol. 62(1):014022, 2019.

[47] [PERRY und MOUROU 1994] M. D. Perry und G. Mourou. **Terawatt to petawatt subpicosecond lasers**. Science, Vol. 264(5161):917–924, 1994.

[48] [SARIKAYA et al. 2018] A. Sarikaya, M. Correll, L. Bartram, M. Tory und D. Fisher. **What do we talk about when we talk about dashboards?** IEEE transactions on visualization and computer graphics, Vol. 25(1):682–692, 2018.

[49] [SHNEIDERMAN 2003] B. Shneiderman. **The eyes have it: A task by data type taxonomy for information visualizations**. In: Proceedings of the Craft of Information Visualization, pp. 364–371. 2003, Elsevier.

[50] [SMITH und CEUSTERS 2010] B. Smith und W. Ceusters. **Ontological realism: A methodology for coordinated evolution of scientific ontologies**. Applied ontology, Vol. 5(3-4):139–188, 2010.

[51] [STREAMLIT 2022] A. Streamlit. **Streamlit: The fastest way to build and share data apps**. 2022.

[52] [STRICKLAND und MOUROU 1985] D. Strickland und G. Mourou. **Compression of amplified chirped optical pulses**. Optics communications, Vol. 55(6):447–449, 1985.

[53] [SWELLER] **Cognitive load theory**.

[54] [TAJIMA und DAWSON 1979] T. Tajima und J. M. Dawson. **Laser electron accelerator**. Physical review letters, Vol. 43(4):267, 1979.

[55] [TOL 2021] P. Tol. **Introduction to colour schemes**. Paul Tol's Notes: Color Schemes and Templates, 2021.

[56] [WARE 2019] C. Ware. **Information visualization: perception for design**. Morgan Kaufmann, 2019.

[57] [WERTHEIMER 1938] M. Wertheimer. **Laws of organization in perceptual forms.** 1938.

[58] [WILKINSON et al. 2016] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne et al. **The FAIR Guiding Principles for scientific data management and stewardship**. Scientific data, Vol. 3(1):1–9, 2016.

Declaration of Academic Integrity

I hereby declare that I have written the present work myself and did not use any sources or tools other than the ones indicated.

Datum: 18.02.2025 .....

(Signature)