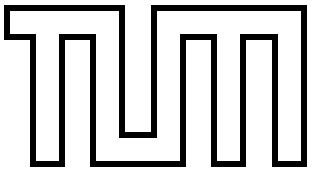


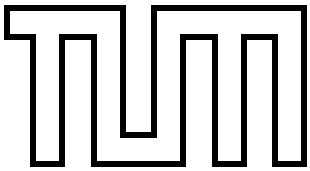
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Development of a Flight Data Archive for MORABA's Sounding Rocket Missions

Florian Koch



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Development of a Flight Data Archive for MORABA's Sounding Rocket Missions

Entwicklung eines Flugdatenarchivs für MORABAs Höhenforschungsraketenmissionen

Author:	Florian Koch
Supervisor:	Univ.-Prof. Dr. Hans-Joachim Bungartz
Advisor:	DLR: Markus Wittkamp, TUM: Manish Mishra
Date:	20.12.2024

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 20.12.2024 Florian Koch

Acknowledgements

I want to thank Markus Wittkamp from MORABA for his guidance and support throughout the entire thesis. He was always available to answer my questions, provide feedback, and share his expertise. I'm especially grateful that he gave me the opportunity to collaborate with the DLR for my thesis.

Furthermore, I want to thank the entire MORABA team for always answering my questions and giving me feedback. By providing a lot of insights into their work, I had a great and fun time working there.

I also want to thank Manish Mishra for taking an interest and acting as an advisor, as well as answering my questions and giving me feedback whenever I needed it.

Abstract

Long-term preservation is an often overlooked subject for space and high-altitude missions. These missions generate large amounts of valuable data in a short period of time, presenting unique challenges. These include managing diverse data types, and capturing the necessary context and metadata to ensure the information remains accessible over time. This thesis addresses these challenges with the insights and context of MORABA, a department of the German Aerospace Center (DLR).

With a focus on preservation, accessibility, and discoverability, established guidelines for scientific data archiving are adapted to the challenges of sounding rocket launches. A prototype is developed to demonstrate how these theoretical concepts can be realized to meet the requirements of a flight data archive. The prototype includes storage entities and a server backend with an API. Furthermore, a frontend application demonstrates how data discovery, access, import, and export can be achieved using the API.

This work combines theoretical and practical aspects of data archiving, presenting a scalable framework for the preservation of flight data.

Zusammenfassung

Die Langzeitarchivierung wird bei Weltraum- und Höhenforschungsmissionen häufig vernachlässigt. Diese Missionen erzeugen innerhalb kurzer Zeit große Mengen wertvoller Daten und stellen dabei einzigartige Herausforderungen dar. Dazu gehören die Verwaltung verschiedener Datentypen und die Erfassung des erforderlichen Kontexts und der Metadaten, um sicherzustellen, dass die Informationen für längere Zeit zugänglich bleiben. Die vorliegende Arbeit befasst sich mit diesen Herausforderungen anhand der Erkenntnisse und Einblicke von MORABA, einer Abteilung des Deutschen Zentrums für Luft- und Raumfahrt (DLR).

Mit Schwerpunkt auf Erhalt, Zugänglichkeit und Auffindbarkeit wurden bestehende Richtlinien zur wissenschaftlichen Datenarchivierung an die spezifischen Anforderungen von Höhenforschungsraketen angepasst. Ein Prototyp wurde entwickelt, um zu demonstrieren, wie diese theoretischen Konzepte praktisch umgesetzt werden können. Der Prototyp beinhaltet Datenspeicher sowie ein Server-Backend mit API. Eine Frontend-Anwendung veranschaulicht, wie Daten über die API gefunden, abgerufen, importiert und exportiert werden können.

Diese Arbeit kombiniert theoretische und praktische Aspekte der Datenarchivierung und stellt einen skalierbares Framework für die Langzeitarchivierung von Flugdaten dar.

Contents

1. Introduction

The journey of Data Storage and Management began thousands of years ago with engravings on clay tablets and stone, evolving over time to the use of papyrus and eventually paper. The development of storage technology's started slow, but increased over time with various revolutionary technologies. The fist revolution was the invention of the letterpress. From there on data storage was possible in a much bigger scale, and storing a big amount of financial data was possible. However working with this data was hard, and information processing was manual. [Gra07]

In 1800, the development of the first automated information process began. The so called punch cards were invented. In the early 1900, a small company called IBM started to produce equipment that can record data on cards. And then the development of new technologies began accelerating increasingly faster. Around five decades later, the first programmable computers were invented in the 1950s, electrical storage of data on magnetic tape was developed. From then on, the journey to relational databases was short. [Gra07]

However with the ever increasing amount and size of data, also these very powerful tools are reaching their limits. New technologies to increase the performance and storage size are continuously developed and are very much needed.

In the context of scientific experiments and tests, data archiving and management is a very important topic. Especially in space and high altitude flights, because of the sheer amount of data that is produced and collected during flights and tests. Many insights can be obtained from this data. Therefore an efficient and easy to use data and archiving solution is the basis for getting the highest value out of the experiments.

However, this was not always the case. The importance of a scientific data archive for space missions evolved after the first missions by NASA in the 1960s after a lot of the experiments were not documented well enough and a lot of highly valuable data was lost. Later they created the Planetary Data System (PDS) in 1989 for the archival of space mission data. ESA has build something similar, the Planetary Science Archive (PSA) for their missions. [BVB+18]

Sounding rocket missions are suborbital flights conducted using rockets designed to carry scientific instruments into the upper atmosphere or near space for short durations. These missions are primarily used for research, technology testing, and atmospheric measurements, providing valuable data before the payload returns to Earth. The high variability in the structure of those rockets and the relatively short duration of these missions, where a lot of data is collected in a high frequency, provide unique characteristics which need to be addressed. Existing approaches for data preservation and long time storage in relation to space are often designed for earth observation missions, making them only partially suited for the unique needs of short-duration space and high-altitude flights.

This thesis aims to address these differences by developing a framework for archiving data from rocket missions and presenting a prototype showing how this framework could be realized. Thereby the implementation is guided by best practices and international guidelines for similar approaches like earth observation. It answers the questions how to develop a flight data archive for high-altitude and space missions with a focus on preservation, usability, accessibility and discoverability, what key metadata is needed to ensure long-time usability of sensor data from rocket missions, and how these theoretical considerations can be practically implemented.

This work was done in collaboration with MORABA, a department of the German Aerospace Center (DLR), whose expertise in conducting sounding missions provide valuable insights into the practical requirements of data preservation.

Part I.

Background and Related Work

2. Technical and Theoretical Background

This chapter provides an overview of the key theoretical and technical concepts related to building a data archive, which is important for understanding its requirements and challenges. It covers foundational concepts of data and data storage, explores various types of databases, and introduces considerations for user interfaces in archival systems. Additionally, the chapter touches on security principles and introduces the concept and challenges of long-term data preservation.

2.1. Storage Concepts

Data storage is the essential aspect of databases. Depending on the size and use of the database, there are different concepts and approaches on how to store data. A key challenge is, that the amount of data is rapidly increasing, the available sensor data from extensive experiments like earth observation and space exploration already exceeds the produced storage hardware per year. Some archives have an annual growth rate between 15% and 65% [MAP19]. This means more data is produced than can currently be stored, leading to a massive challenge for data management systems and requiring the development of new concepts and technologies every year.

There are many concepts and approaches to data storage for many different needs and use cases. These include cold versus hot storage, active versus static storage, and on-premise versus cloud storage. Each method has unique characteristics and advantages that fulfill the requirements of a particular storage solution.

2.1.1. Cold versus Hot Storage

The first discussed concept is cold versus hot storage. Cold storage archives are archives where most of the data written into the archive is read very rarely [MAP19]. The data is stored mainly for long-term preservation. This has, of course, significant impacts on the design of the archive because features like querying and read throughput are less critical, because these mechanisms are not used very often. Therefore, low-cost storage hardware can be used with a cheaper price-per-size ratio. This kind of storage hardware, like tape, for example, can store a large amount of data with the downside of having limited throughput, but because the data is rarely accessed, this is less important. [MAP19]

On the other side, there is the concept of a hot archive. Where all the data has to be available on-demand, often, this data is accessed by multiple users, and therefore, the throughput has to be higher compared to a cold archive. Multiple user lead to concurrent read and write operations, which the database has to handle. Furthermore, real-time operations might be

needed. Therefore, the requirements for a hot data archive are much stronger and more difficult, especially performance is significantly more relevant, which is why more expensive hardware is used for hot storage solutions. The more expensive hardware provides higher throughput and overall better performance, which is needed when the database is accessed more extensively. [MAP19]

Therefore, the main factors that differentiate between a cold and a hot storage archive are the access frequency and the needed retrieval speed. However, for some applications, the data volume makes it challenging to use a hot storage concept, and certain trade-offs have to be found. [MAP19]

2.1.2. On-premise versus Cloud

Another aspect of designing a database or archive is whether an on-premise or a cloud solution is needed.

The main difference between these two options is that with an on-premise solution, everything, including hardware, software, and applications, is managed internally, meaning no third party is involved. For a cloud storage solution, on the other hand, everything is managed by the cloud provider. There are different aspects to consider when deciding whether to host on a cloud service or not, such as flexibility, deployment, cost, maintenance, and security. [GSB21]

When talking about deployment and maintenance, cloud solutions are the easier option because the cloud provider takes care of these aspects. In contrast, an on-premise solution is managed by the company itself. So, with a hosted database, the user does not have to worry about backups, upgrades, and maintenance. Most of the time, the cloud solution is also cheaper because the hardware costs alone are more expensive than the cloud subscription. Additionally, the development and maintenance costs come on top, whereas with cloud solutions, there are subscriptions, which are only paid for what is used. Also, cloud-hosted databases are preferable in terms of flexibility because up-scaling (increasing the resources of the database) and down-scaling (decreasing the resources of the database) can be quickly done through the cloud provider. This means the resources can be easily up-scaled during periods with high demand and down-scaled in periods with low demand. It is much more difficult for on-premise solutions because the hardware must be bought and managed. Therefore, it is more expensive and more complicated. [GSB21]

However, the big downside of cloud solutions is the security aspect and possible vendor lock-ins. Once data is stored externally, it is not in the company itself on company storage hardware. It is hard to tell who else has access to this data. Of course, it is claimed by the cloud providers that only the company that owns the data can access the data, but this is not verifiable in any way. Therefore, it is also important to consider which data gets stored. If it is sensible data like personal or military data, the company often does not allow it to store this data in externally hosted clouds. Another significant downside of cloud storage solutions hosted by a third party is the danger of a vendor lock-in. [OMST14]

This describes a situation in which a customer becomes dependent on a single vendor,

making it difficult to switch to another vendor without incurring significant costs, risks, or disruptions. This is one of the major problems related to cloud providers. These often try to lock in the customer on purpose by designing systems and architectures incompatible with other vendors and licensing software under exclusive conditions. [OMST14]

Ultimately, the decision between an on-premise and a cloud solution should be guided by a thorough assessment of the company's needs, resources, and goals. Careful consideration of these factors can lead to an informed decision that aligns with the company's long-term strategy.

2.1.3. Active versus Static Storage

The concept of active storage versus static storage is related to the concept of hot storage versus cold storage, which shares many similarities but with a different distinguishing factor. While for cold and hot storage archives, the distinguishing factors are access frequency and retrieval speed, the difference between active and static storage systems is the active modification. [MAP19]

For static storage systems, data gets written in the archive once and does not change again. So, it is in a "frozen" state. However, it does not state anything about the access frequency. So, hot static storage and cold static storage systems exist. On the other hand, active storage systems are where the data inside the system is currently being worked on or used extensively. Therefore, it is often changed and modified. This means that different user interfaces must be implemented, such as for uploading and modifying data. Also, version control might be required for active storage systems. Additionally, different operations might be performance-critical for usability, depending on the concept type. For example, ingestion operations are more often performed in active than in static storage systems. [MAP19]

To conclude, the choice between these storage options depends on various factors, including data accessibility requirements, budget constraints, security considerations, and especially the specific use cases of the storage solution. Each approach has its benefits and trade-offs, making it crucial for organizations to carefully evaluate their needs and select the most appropriate data storage strategy.

However, there is no hard distinction between the different approaches. It is important to note that there are many solutions that can adapt and combine different approaches to extract the best possible fit for a particular application, providing a sufficient level of flexibility.

2.2. Data

This section will present important aspects of data when planning and developing a data management system. This includes determining what kind of data needs to be stored and the amount of data to store. Understanding these aspects is crucial for finding the right database and developing an efficient and effective data management strategy. Generally, (Big) Data is defined through the five V's: Variety, Volume, Velocity, Value, and Veracity [CLR23]. These five concepts will be presented in the following.

2.2.1. Data Variety

The first aspect of data to consider is data variety. This term refers to the diverse types and aspects of data crucial for analyzing situations, conducting scientific experiments, or making business decisions. Different databases or storage solutions must be found depending on the type and how many different types of data should be stored. Which often proves to be a challenge. The various types of data can be categorized into three distinct kinds. [LH19]

The first one is *relational* or *(fully) structured data*. This is the classic data stored in relational databases like PostgreSQL or MySQL. The data is represented as tuples forming relations and are uniquely identified by a key formed out of one or more values of one tuple. The typical approach for retrieving and storing this kind of data is the Structured Query Language (SQL). [LH19]

The second one is *semi-structured data*. It is categorized between structured and unstructured data, possessing some structure, such as key-value pairs, but not the rigid organization of structured data. This data type is often represented in JSON, HTML, or XML formats. Often, this type of data is stored in NoSQL databases. [SSSF09]

The third and last data form is *unstructured data*. This data is commonly referred to as data that cannot be stored in a row-and-column approach. Typical unstructured data are images, videos, or whole documents in, for example, PDF format. Because of this data's unstructured nature, it is mostly stored as files on servers. [SSSF09]

2.2.2. Data Volume

However, not only the variety in data types is a challenge for developing a data management system. The volume and size of data in the modern world is a massive challenge. In addition, a good system should be functional for several years or even decades. Therefore, it has to be scalable and maintainable. Furthermore, when looking at the explosion of data growth in recent years, scalability is a huge factor.

To put this into numbers, the whole of humanity produced 5 exabytes (10^{18} bytes) of data until 2003. In 2013, this amount was produced in two days. Moreover, in 2013, the total amount of data was expected to double every two years. [SS13] This did not hold entirely true. In 2024, there were 402 exabytes of data produced. This is 80 times the amount of data produced until 2003. However important to note is that only around 2% of this data was stored for a longer period than one year [Dua24]. This low percentage of stored data highlights the need for efficient data management systems to identify and retain the most valuable data. So, there is a massive difference between produced and stored data, which is important to consider when creating a database management system.

Figure 2.1 shows the annually generated data from 2010 to the estimated generated data in 2025. This figure puts into perspective how drastically the generated data is growing. More devices are producing more data, more extensive data like images and videos are getting a higher resolution and are, therefore, increasing in size. For that reason, while developing a concept for a data management system, the growth of the volume of data is an important issue that must be considered.

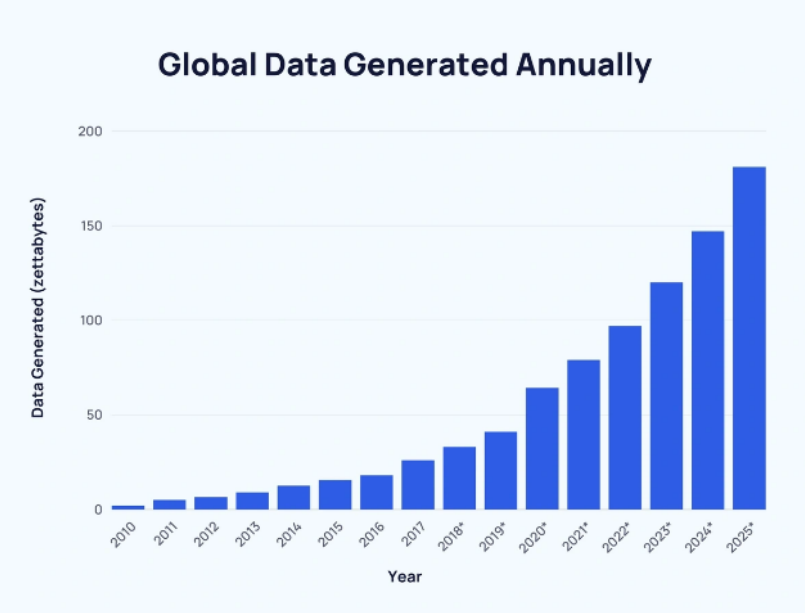


Figure 2.1.: The Annual Generated Data from 2010 until 2025 [Dua24]

2.2.3. Data Velocity

Data velocity refers to the rate at which data is generated and processed. This concept is closely tied to the performance requirements of systems, particularly those designed for real-time operations. In real-time systems, where data is generated, collected, distributed, and processed almost instantaneously, the performance demands are significantly higher and more critical compared to systems where such immediacy is not required. The optimization of real-time systems to efficiently handle the rapid data flow is a crucial factor in meeting the performance criteria essential for timely decision-making. [SS13]

2.2.4. Data Value

Data value refers to the importance and utility of data in generating insights and making informed decisions. The value of data is a key aspect in deciding whether the data should be stored and for how long. Data with high value is more likely to be retained, while data with limited or decreasing value may not be worth storing over a long time. As previously noted, only 2% of generated data is stored for more than a year. Therefore, evaluating data value is crucial for developing an efficient storage strategy, as it helps avoiding unnecessary data retention, which can impact the systems' data volume and overall performance. [CLR23]

2.2.5. Data Veracity

Data veracity refers to the quality and reliability of data, encompassing the information's accuracy, trustworthiness, and overall integrity. High data veracity is crucial because it directly impacts confidence in the insights and decisions derived from the data. Unreliable or inaccurate data can lead to flawed conclusions and poor decision-making. As a result,

evaluating and ensuring data veracity is essential in data management processes, as it influences the credibility of the data and the effectiveness of any subsequent analysis or application. [CLR23]

2.3. Databases

Databases are the backbone of modern data management, providing structured ways to store, retrieve, and manipulate data efficiently. As organizations generate and handle increasingly complex and voluminous data, choosing the right database system becomes more critical. There is no single solution that fits all problems. It is influenced by the specific attributes of the data itself, often derived from the 5 V's of data introduced before. Each of these dimensions influences the choice of database, as different types of databases are optimized to handle particular data challenges. For example, data variety, which refers to the diverse types and formats of data, plays a significant role in determining whether a relational, NoSQL or other specialized database is most suitable. Understanding the interplay between the five V's and database capabilities is critical for designing a data management system that effectively supports organizational needs.

This subsection presents various database concepts handling the basic data types.

2.3.1. Relational Databases

The relational database, the classic database, was invented in 1970 by E.F. Codd [JPA+12]. In relational databases, the data is stored in tables consisting of rows and columns. The key feature of these databases is the use of relations to connect the tables, forming a structured and interconnected system. Each row in a table describes a data point with different values of different categories, shared by every data point in a table. These categories are the columns.

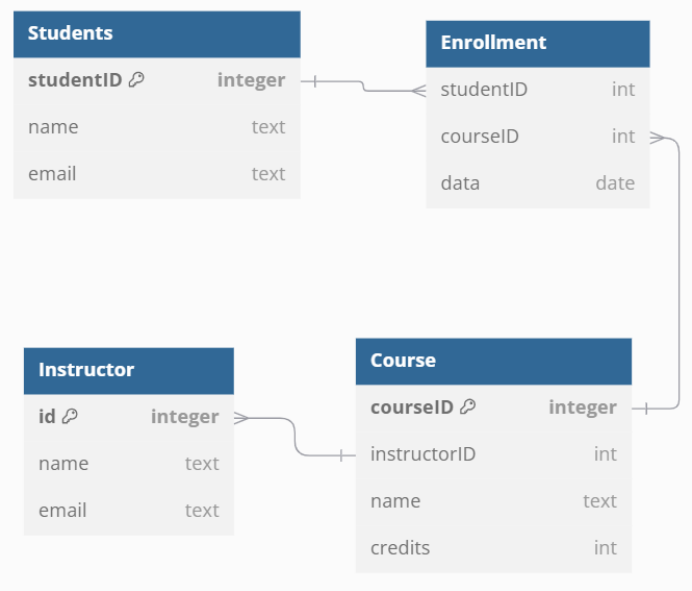


Figure 2.2.: A Example for a simple Schema of a Relational Database

Figure 2.2 shows a simple but typical database schema. This example consists of four tables: Students, Course, Enrollment, and Instructor. Each table has a primary key and a unique identifier, ensuring that every record is distinct. For instance, studentID in the Students table and courseID in the Course table are primary keys.

Foreign keys establish relationships between tables. In the Enrollment table, studentID and courseID are foreign keys referencing the Students and Course tables, respectively. This connects students to their enrolled courses. Similarly, in the course table, instructorID is a foreign key pointing to the Instructor table, showing which instructor teaches each course.

Relational databases are built around relationships, which can be one-to-many (e.g., one instructor teaches many courses) or many-to-many. The Enrollment table is an example of a many-to-many relationship, linking students to courses without duplicating data. This concept is central to database normalization, a process that organizes a database to reduce redundancy and dependency. It ensures that data is stored efficiently and accurately and that integrity constraints, such as primary and foreign key rules, can be applied to prevent invalid or duplicate entries, thereby ensuring data consistency.

Relational databases are particularly powerful when it comes to querying and retrieving data. The Structured Query Language (SQL) allows users to extract information by combining data from multiple related tables through 'joins'. A join is an SQL operation that combines rows from two or more tables based on a related column between them. These joins allow access to data stored across different tables without redundancy. For example, SQL can be used to join the Students, Enrollment, Course, and Instructor tables to find all students enrolled in a specific course and the instructor who teaches that course. This query would pull information from each table based on the relationships established by foreign keys, providing a cohesive result that shows which students are enrolled in which courses and who teaches them.

By leveraging joins, users can perform complex queries that provide deep insights into the data without duplicating information across tables. This allows databases to remain normalized and efficient, preventing unnecessary data storage. Additionally, SQL queries can filter, sort, and aggregate data, allowing users to perform anything from simple lookups to complex analytical queries while maintaining the integrity and consistency of the underlying data. This ability to handle complex data relationships while ensuring data remains accurate and up-to-date is one of the key strengths of relational databases.

However, there are also certain disadvantages to relational databases. One major drawback is that they are not well-suited for handling unstructured data, such as images, videos, or large text files, which are becoming increasingly common in modern applications. Relational databases rely on a structured schema, making storing and managing flexible data difficult. Additionally, relational databases can provide relatively low scalability. These databases can struggle with performance issues as data grows unless carefully distributed across multiple servers. [JPA+12]

2.3.2. NoSQL Databases

NoSQL databases are becoming increasingly popular, offering a flexible alternative to traditional relational databases. One of the key differences is that NoSQL databases do not

rely on tables or use SQL for querying. Instead, they use different models for organizing data, which allows for greater flexibility and scalability, especially when dealing with large volumes of unstructured or semi-structured data. There are various NoSQL databases, each categorized based on how they organize data, each suited to different applications. [JPA+12] The most popular categories of NoSQL databases are:

- **Key-value stores** organize data into simple tuples consisting of a key and a value. The key, typically a string, uniquely identifies each tuple, while the value represents the actual data. The value can be anything from an integer to a more complex object, allowing key-value stores to be highly flexible and loosen the rigid requirements of fixed data structures in relational databases. This simplicity makes key-value stores ideal for applications that need fast lookups and can tolerate a flexible data structure, such as caching systems or real-time analytics.[JPA+12]
- **Document Stores** organize data as collections of documents, where each document is a self-contained unit of data, often formatted in JSON, BJSON, or XML. Unlike relational databases, document stores do not require a fixed schema, allowing each document to have its own structure. A document typically contains key-value pairs, but the values can also be complex, such as arrays or nested documents, making this type of database highly flexible. [JPA+12]
- **Graph Databases** are schema-less and use a structure of nodes and edges to represent data. In this model, nodes represent entities, such as people, products, or locations, while edges define the relationships between those entities. Each node can store various attributes, and edges describe how nodes are connected, often including details about the nature of the relationship. This structure makes graph databases well-suited for applications requiring complex, interconnected data analysis.

Next to these three NoSQL categories are also other, less popular categories like column-oriented databases, Object-oriented databases, Grid databases, and XML databases. [JPA+12]

2.3.3. Multi-model Databases

Relational and NoSQL databases are often optimized for handling one specialized form of data, such as structured data in relational databases or unstructured data in certain NoSQL systems. However, many modern applications require managing multiple kinds of data within a single system, which can be challenging using a single database type. This has led to the introduction of multi-model databases, which provide a unified platform capable of handling multiple data models—such as relational, document, graph, and key-value—within the same database. By supporting various data types and structures in a single system, multi-model databases offer greater flexibility and simplify data management. [LH19]

In Figure 2.3 is an example from *Multi-model Databases: A New Journey to Handle the Variety of Data* by Jiaheng Lu and Irena Holbová [LH19], where different kinds of data are needed. In this example, the data of an e-commerce platform is portrayed, where Customer information, such as names, is stored in a relational table. The relationships between customers, such as referrals or social interactions, are stored as graph data, where each customer is represented as a node, and their connections are edges. Order data, including order IDs

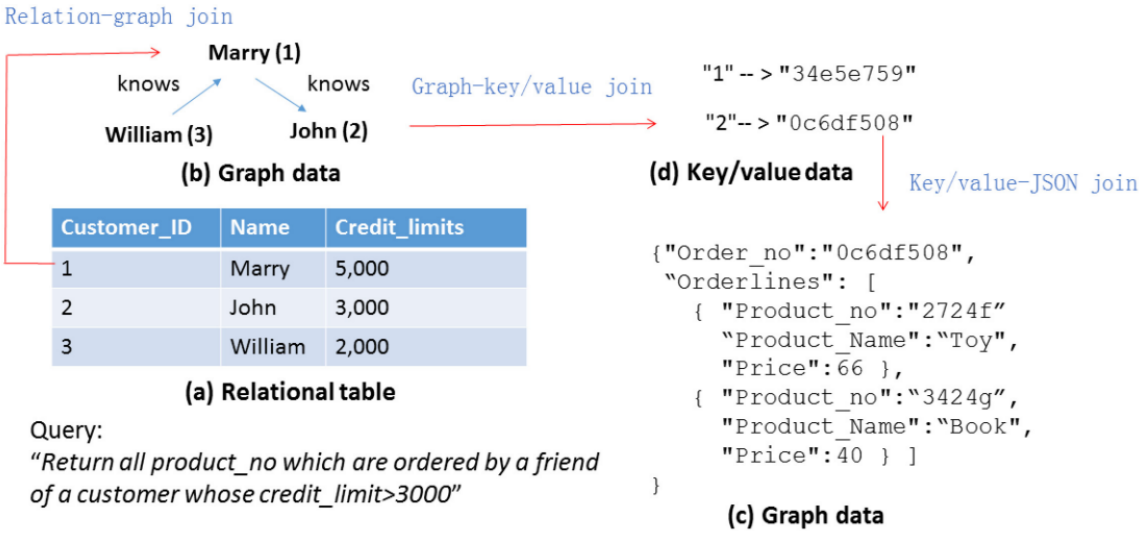


Figure 2.3.: Example of Different Data Types for a E-commerce Platform [LH19]

and purchased items, are stored in JSON documents. Furthermore, the relationship between customers and their orders is managed in a key-value store. This data structure presents a challenge for traditional databases. [LH19]

On this data, a typical query could be *"Return all product numbers ordered by a friend of a customer whose credit limit is greater than 3000"*. There are three approaches to how this can be solved. The first approach involves using four separate databases, each optimized for a specific data model (relational, graph, document, and key-value). While this ensures efficient storage for each type, it creates significant installation, administration, and integration challenges across multiple systems.

The second approach is to convert all data into one format, such as a relational database. This simplifies management but can negatively impact query performance, especially for unstructured or complex data like customer relationships or order details.

The third approach leverages a multi-model database, which supports all data types within a single system. This solution offers the benefits of specialized data models while simplifying administration, making it the most flexible and efficient option. Examples of such a database are ArangoDB and OrientDB. [LH19]

However, there are drawbacks to the multi-model database approach. First, these databases typically use their own query language, which requires additional learning to effectively work with the system. In contrast, using a relational database ensures familiarity with a widely-known query language, such as SQL, and access to a broad range of APIs. Additionally, general query languages designed to work across different database models often lack the performance of native, specialized query systems, leading to slower or less efficient data retrieval compared to dedicated databases. [LH19]

Ultimately, the choice of a database, including whether a multi-model database is the right choice, depends on the specific data needs and operational environment. Multi-

model databases offer flexibility and simplified management by integrating various data models. Despite the challenges, such as learning new query languages and potentially lower performance compared to specialized databases, the benefits of a multi-model database make it a valuable option. On the other hand, specialized databases can optimize performance for specific data types but require managing and integrating multiple systems.

2.3.4. Data Warehouse

While multi-model databases provide the flexibility to handle different data types and models within a single system, they primarily focus on operational efficiency and versatility in managing diverse use cases. However, when the goal shifts from operational processing to in-depth analysis and strategic decision-making, another specialized type of database can be an option: the data warehouse.

Introduced in the late 1980s by IBM researchers Barry Devlin and Paul Murphy, Data warehouses were created to organize and manage large volumes of data, making it easier for businesses to analyze and use that information to support important decisions. They gather data from various sources, structure it, and provide a central place for businesses to perform analysis, helping leaders make more informed choices based on comprehensive insights. According to Inmon's widely accepted definition, a data warehouse is a subject-oriented, nonvolatile, integrated, and time-variant data collection intended to support management decisions. They are used in various sectors, including production, business, monetary administration, logistics, etc. [NM22]

A core process of data warehouses is the so-called Extract, Transform, Load (ETL) process. This process describes how data gets into a data warehouse. The key components are *extraction*, which describes collecting data from various systems. The *Transformation* step filters, cleans, and transforms the extracted data into the standard format of the data warehouse, and *loading* imports the transformed data into the warehouse. The tools that carry out this process also perform other tasks like updating the data in the warehouse and maintaining the metadata. [NM22]

Since their introduction in the 1980s, data warehouses have been the focus of extensive research and development, gaining increasing attention and importance, especially in the era of big data. The future outlook remains strong, with a report by Market Research Future predicting significant growth in the Data Warehouse as a Service (DWaaS) market, projected to reach USD 7.69 billion by 2028, with a compound annual growth rate of 24.5%. [NM22]

However, the rapidly increasing volume of structured, semi-structured, and unstructured data brings significant challenges for traditional data warehouses, leading to three major issues. First, due to their architecture, data warehouses can only address predefined requirements, making them less adaptable to the fast-paced changes in the digital world. Second, valuable and potentially important information may be lost during the ETL process, which can impact the quality and completeness of the data. Furthermore, as data volumes continue to grow, the costs associated with maintaining and improving performance in data warehouses are growing exponentially. [RZ19] These issues led to the development of a new management system called Data Lake.

2.3.5. Data Lakes

To address the challenges of the fast-growing amount of data, the concept of a Data Lake (DL) was introduced as a more flexible solution for big data. Unlike data warehouses, which store transformed and structured data, data lakes ingest raw, heterogeneous data without prior transformation, allowing it to be processed as needed. According to J. Dixons explanation: *"whilst a data warehouse seems to be a bottle of water cleaned and ready for consumption, then "Data Lake" is considered as a whole lake of data in a more natural state"* [NM22].

In essence, a data lake is a storage system that holds data in its raw form, whether structured, semi-structured, or unstructured. The responsibility for transforming and processing the data falls on the user, giving data lakes a more flexible architecture. This lack of strict organization reduces overhead and improves performance, as data does not need to be transformed upon entry but only when required for analysis.

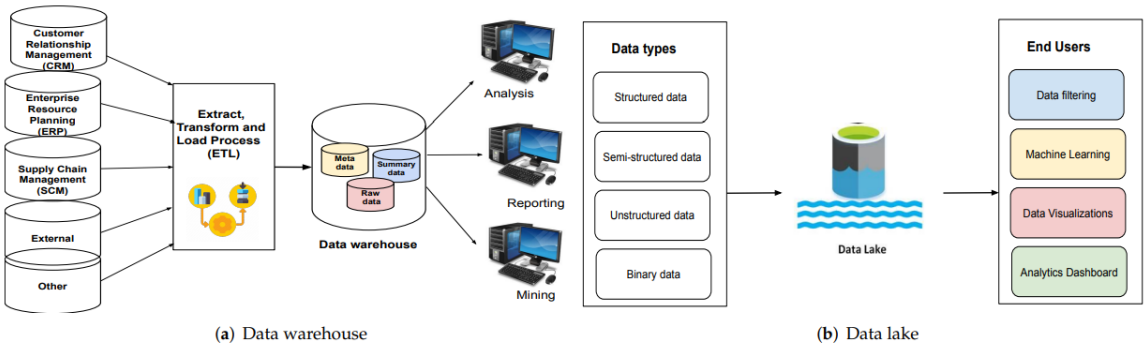


Figure 2.4.: Difference between Data Warehouse and Data Lake [NM22]

The difference between a data warehouse and a data lake can be seen in Figure 2.4. On the left side is the outline of the architecture of a data warehouse, and on the right side of a data lake. Both get data from different sources. However, the data lake also ingests different data types. At the data warehouse, before being put into the actual database, the data runs through Extract, Transform, Load (ETL) tools, which extract data from various sources, transform it into a consistent format, and load it into the destination. In contrast, in the data lake, the data just gets thrown into the system. After that, analysis, reporting, or mining can be done directly in the warehouse because the data is already cleaned and filtered. To work with the data of the data lake, the end user has to process the data but can decide how he wants to process the raw data and use it for machine learning or simple visualization.

2.3.6. Time Series Databases

Next to the different types of databases presented, there are also various specializations of databases for specific use cases. A time series database is one of these use cases, particularly useful for a flight data archive. A lot of flight data consists of time series data, where one data point consists of a timestamp and an associated value, for example, from a sensor.

Time series databases are optimized for such timestamp data. These optimizations are focused on efficient storage and high-performance queries. [GLVF21]

However, there are different approaches to building a time series database, like building an extension for an existing database or developing a completely new one. In the following, three different databases with different approaches are presented.

MongoDB

MongoDB is one of the most popular NoSQL databases. It is an open-source document-oriented database that stores data in a JSON-like format supporting nested sub-documents and offering high flexibility for expressing relationships. MongoDB consists of collections, which are like a table in relational databases. There exists a specialized collection for time series data. This collection is optimized to reduce disk usage and improve query times for timestamp data. This is achieved by using time as a clustered index. Additionally, it provides a powerful querying language that includes possibilities for range-based queries and other options specialized for time series data. [VCL23]

The advantage of MongoDB is that by being structured in the JSON format, metadata to each entry is easily addable. The disadvantage is that while providing a powerful query language, it is not possible to use SQL, with which many people might be familiar. So, a new query language has to be learned.

InfluxDB

InfluxDB is an open-source database developed for time series data. The first two versions were developed in Go, and the third and current versions is developed in Rust. Therefore, InfluxDB is naively optimized for time series data with high ingestion rates and efficient query performance. It uses InfluxQL, which is a SQL-like query language. [Inf24]

The disadvantage is that while efficient, its storage requirements can be higher than other similar databases [MFP+19]. Additionally, it cannot be integrated as simple as extensions for common SQL or NoSQL databases because it is built only for time series data, so another database for different data is needed.

TimescaleDB

TimescaleDB is a open-source extension for the open-source database PostgreSQL. It optimizes time series data and provides fast reading and complex queries. The main advantages are high data reading rates for large-scale data, a high query performance, and time-oriented features that provide a huge performance improvement for time-oriented queries. The tables, which store the time series data and use the TimescaleDB extension, are called hypertables. These hypertables partition the content into chunks. The chunk size can be defined over time and space. The huge advantage compared to NoSQL time-series databases is that it is integrated seemingly into normal PostgreSQL and SQL queries can be used on the TimescaleDB hypertables.

To summarise, all three databases can store time series data. However, depending on the use case, each of the three could be suited better. A study on Hefei Advanced Light Facility (HALF) data archiving system was made by Chen et al., where different time series

databases were tested, including the three presented above [CLZ+23]. In figure 2.5, a result of a reading test can be seen. Thereby are on the x-axis the different databases and on the y-axis the according query datapoints per second (QDPS), so how many data points can be retrieved from the database per second. Two tests with the same amount of data were made, one with newly ingested data and another with data stored in the database for longer. [CLZ+23]

It is observed that TimescaleDB outperforms the other databases for both newer and older data by a margin. Most databases perform faster on new data than on old data due to differences in data storage methods. The study also highlights that TimescaleDB's software ecosystem is significantly better than that of the other databases, as its integration with PostgreSQL makes it compatible with many visualization and analytical tools, facilitating integration within a large-scale data processing environment. [CLZ+23]

However, the database choice highly depends on the use case, personal preference, and the existing software environment.

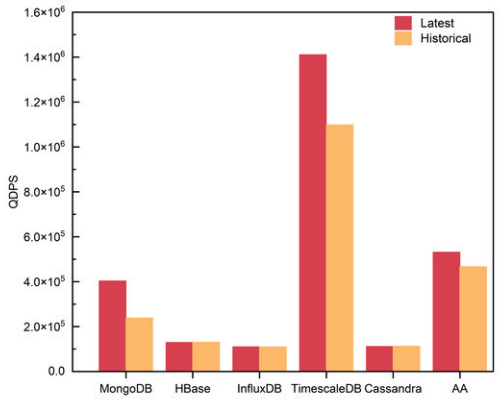


Figure 2.5.: Comparison of the reading Speed of different Time Series Databases from [CLZ+23]

2.4. User Interface

In this section, a short look at user interfaces for archives is done. While the user interface is not the main goal of this thesis, it plays an important role in a data archive. It is the connection between the user and the archive, which is critical for the usability of the archive and plays a decisive role in deciding whether the data archive is used by the target user group.

Web-based data portals, which offer powerful search capabilities and on-demand data retrieval, are widespread. However, there is yet to be a consensus on how such a tool should look, and the various interfaces and search engines can have substantial differences. [SPT19] Search engine interfaces are built for users, and therefore, it makes sense to analyze the behavior of users to gain insights and analyze essential factors that are important when

building such interfaces. The DLR followed this approach to develop generalized recommendations for search frontends of data portals. The DLR maintains the German satellite data archive consisting of large earth observation data. Section 3.1.2 presents this data archive in more detail. This data archive also includes a search frontend for various user groups to access the data. To gain insights into the user behavior of this earth observation data portal, log files of the search request were collected and analyzed over six months. To find data in the archive, users of that specific data portal can restrict the results with five constraints. First, they can browse by collections, which are sets of related data. Then, they can use a map interface to restrict the results to certain spatial regions and temporal restrictions. The fourth constraint is a common keyword search, which matches the provided keyword with the data set's content. And the last constraint is a restriction by type. [SPT19]

The insight they gathered was that also the interface provided a specialized spatial restriction interface, the users used the keyword restrictions for the same purpose. This resulted in poorer results because the keyword search was not designed for this purpose. Generally, there were some issues with the keyword search, which can be summarized by typographical issues, such as spelling and wording issues, abbreviations issues, and semantic issues, influenced by the different backgrounds of user groups.

To summarize the insights gathered by the search log analysis, implementing a keyword search can be challenging because various issues emerge. To address these issues, an extensive metadata collection is needed because the data gets queried based on its metadata. Furthermore, a more detailed metadata description of the data helps the user to find the data. Additional techniques like levensthein distances and code lists should be implemented to ensure a good query performance. [SPT19]

2.5. Security

This section gives a short overview of security in database systems. The three most important security aspects in database systems are authorization and access control. Authorization describes the specification of access rights, so which user or group role can perform what kind of operation, like inserting or retrieving data on which resource. Access control enforces these access rights and controls who gets data access. Due to the various database concepts presented in Section 2.3, general solutions existing for all database models are impossible. Different models can have different access control requirements depending on the data inside the database (structured, unstructured, etc.). A database system has to cover three core requirements. These are confidentiality, integrity, and availability. Confidentiality refers to the protection against unauthorized access. In contrast, integrity refers to the protection against unauthorized and improper data modification inside the database, and availability describes the protection against the unavailability of hardware and software. [MAHK23]
Out of these requirements, Mohamed et al. [MAHK23] defined more detailed requirements for data store solutions independent of their database models, which are:

- Varying Granularity of Protected Data Objects
- Content-Based Authorization
- Context-Based Authorization

- Custom Authorization
- Separation of Concerns

Varying granularity describes the requirement that access control has to be implemented on different levels of the data schema. For relational models, for example, it should be definable for certain user roles to have access to whole tables or only certain columns, etc. For relational models, this is relatively simple because the database models have a fixed schema. However, for example, NoSQL models do not have a fixed schema, which provides a bigger challenge. [MAHK23]

Content-based authorization describes that access control not only considers the structure of the data but also depends on the semantics of the data. An example would be that only a certain period of a rocket flight is confidential for certain user groups, and based on the timestamp, different user groups have access to that data. In contrast, context-based authorization decides if the user has access to the database, not depending on the data he wants to access, but rather his context. For example, the user might only have access to the database if he is in the office using an allowed computer or network. This is, however, not the opposite of content-based authorization, and these different kinds of authorization can be combined. [MAHK23]

Custom authorization describes the problem of some databases only providing standard, built-in access control mechanisms. However, custom-made access controls provide much more flexibility. This is a prerequisite to fulfill the second and third requirement. The last requirement is the separation of concern. This describes that access control should be independent of the underlying storage solution to offer better scalability and maintainability. [MAHK23]

2.6. Long Term Archiving

After the previous section about the basics of data and databases, the next section will cover another important aspect of building a data archive: long-term storage. The choice of a database management system is not solely based on the type of data it needs to store but also on the specific requirements of an archive. Different storage systems are designed for different phases of data storage. Whether for managing live data, recently completed projects, or storing data for a long time.

Long-term archiving has challenges that are different from conventional data management. Defining what qualifies as "long-term" can be difficult, as the term depends on the context and purpose. The Open Archival Information System (OAIS) provides a practical definition, describing long-term as a period "long enough to be concerned with the impacts of changing technologies, including support for new media and data formats." [SRMP02] When considering building a data archive, which should last, the first ideas and concepts are about this archive's technical and hardware side. So how to ensure that no data is lost, get data into the archive, and get data out of the archive. Maybe also storage technology, which data types have to be stored, and which databases should be used depending on the

types. While this is an essential and critical part of building such an archive, it does not represent the complete challenges. Another challenging task is encountered when taking a step back and looking at the problem aside from the technical challenges. The goal of the archive is to have usable, interpretable data in future decades. This is a time when most of the people who were actively working to produce the data are not in the company anymore. Additionally, most of the hardware and software used during production is likely already replaced by newer, better ones. Especially in a field as complex as space flights, it is crucial to know how the data was created and how the system used at that time worked. It comes down to simple things like what calibrations were used because the data becomes more or less incomparable without them.

Long-term archiving includes storing data and ensuring it is safe, comprehensible, and accessible even as technology develops and changes. It is particularly important in sectors like scientific research, healthcare, and public administration, which produce vital, high-value data.

Compared to active or short-term storage solutions, long-term archiving necessitates special requirements such as data format sustainability and secure storage. It often requires specialized systems and strategies for data migration, redundancy, and metadata preservation to ensure that data will be valuable and accessible in the future.

In addition, more than preservational reasons exist to store data over a longer period. There are also legal and financial reasons. Not so much for scientific data, however there are governmental regulations that define preservation periods for medical, accounting, and labor purposes that can be used in legal or financial disputes. [VS14]

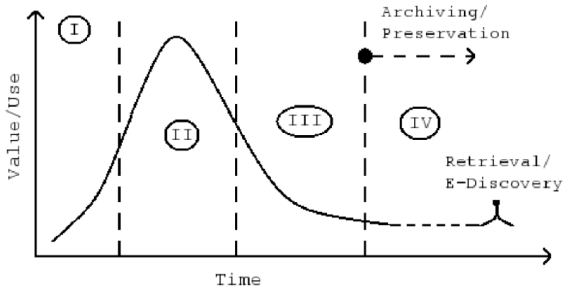


Figure 2.6.: Information Lifecycle curve from [Tal13]

Figure 2.6 shows a visualization of the data and information life cycle. On the axis is the value of the information over time. In the first step, data gets collected, and in the second step, the short term, the data is the most valuable for ongoing operations and procedures. Over time, it slowly loses value for the active operations until, at one point, the value for active projects is static, and it serves mainly as preservation. This is the point in the figure marked as step four, where an archive comes into play. From then on, there might be some value spikes if somebody uses the old data for new insight through the archive.

2.6.1. Data Preservation

As shown in Section 2.2.2, the data volume has increased at a significant rate over the past years and decades. With an increasing tendency. To be able to manage this amount of data, it is important to organize the preservation of the data, so how and for what time the data should be stored. Especially the storage of unstructured data is one of the main causes of data excess in organizations and, therefore, one of the main reasons for the waste of resources [VS14]. So, inefficient data preservation is not only a problem regarding information loss but also a financial factor that should not be underestimated.

In general, data preservation is the ability to ensure that digital data being stored today can be read and interpreted many years from now. This can be interpreted in two different ways. The first is bit preservation, which means the data can be read from the storage medium in the future. The second is logical preservation, which ensures that the information is understandable and usable, regardless of the actual bits on the medium.

While bit preservation is vital for long-term preservation, logical preservation is as important to make sense of the data and keep all the information stored in the bits. A lot of research and work has been done in the field of bit preservation, but not so much in logical preservation. Therefore, NASA developed the Open Archival Information System (OAIS) reference model [SRMP02].

The OAIS reference model is a comprehensive system that outlines the roles of systems and people in preserving specific data over a long period. To ensure both bit preservation and logical preservation, metadata is stored alongside the actual data. This metadata, which includes context information, reference information, and fixity information, plays a crucial role in preserving the data's meaning and context. The OAIS model also defines several data migration types, such as refreshment and replication, to ensure the safe storage of data, independent of the medium. A more detailed look into the OAIS model and similar projects is given in Section 3.2. [SRMP02]

The main reasons for data loss, without taking unusual reasons like fire or natural catastrophes into account, are aging of the storage medium itself, changes in the application software, the description of bite streams being lost, and therefore, the meaning of the data is lost and overwriting files, which is more a user error, but one that occurs often and results in the loss of data. [HP12]

In the end, data preservation is highly dependent on the use case. For instance, there is no need to store data for decades for some use cases because the information is decreasing over the years. However, for other use cases, data collected over 100 years ago might still be very valuable. Therefore, there cannot be a one-size-fits-all policy or process on how to preserve the information, especially for logical preservation.

2.6.2. Data Types

An important aspect to consider for data preservation are data types. These might change with the technological evolution over years or decades. To enable consistency across a data archive, standard data sets are often defined. Defining such a standardized data format set can be challenging because many different data formats exist for different types of data, all with advantages and disadvantages. In general, some principles for file formats should be met in the context of long-term data preservation. The format should be simple to describe, understand, and implement. It should not depend on specific hardware, an operating system, or software. In addition, the format should be robust against single points of failure. Four basic types of file formats are considered. These are text, image, sound, and video formats [Bar07].

For the preservation of images, for example, a compromise between storage size and image quality has to be made. Storing images with a high image quality is expensive in terms of storage size. However, when using a file format that reduces the image size, often image quality suffers. For the long term, the file format TIFF was one of the most popular open formats in the preservation community. TIFF is a lossless format which can be edited and saved without losing quality. JPEG2000 is another format that has gained popularity. It offers a compromise between storage, image quality, and compression. Some reports describe JPEG2000 as the best-suited image format for archiving. However, which format is best for a specific use case is highly dependent on the application and use case of the images and if image quality is important. [EI19]

2.6.3. Data Preservation in Science and Earth Observation

The data preservation and management of scientific data, especially earth and space observation, is one of such use cases. In this area, massive amounts of data, like sensor data, are produced quickly. This data does not only have to be managed and stored, but the required metadata to make the data useful and comparable also has to be defined and stored. Moreover, with the development of ever-new analysis tools and possibilities like machine learning, which can manipulate and explore massive amounts of data, the information we can extract from the recorded data can significantly increase over the years. Therefore, it is essential to ensure the logical preservation of this data because we do not know yet which information could be extracted from this data in the future. [AM20]

However, the main challenge is not only the volume of the data but also the variety and the diversity in format and type. Data is recorded on different kinds of devices and comes in different types and formats. Therefore, it is challenging to construct a data management and archive that works for all data. To enumerate the tasks of such a system, it has to do reformatting, storing, sharing, transferring, transcription, and many more. [AM20]

Particularly for scientific data collected by space missions, long-term archiving can be challenging. Nevertheless, at the same time, the mission data is the only thing left after a mission has ended, and often, it is challenging and expensive to reproduce the data collected by a space mission. Therefore, it is vital to store that data properly. One big challenge is that the duration of a space mission compared to a computer generation is a significant

mismatch. One space mission often includes several computer generations, so most of the time, outdated hardware is used. To show one example, the Voyager mission was started in 1977 and operates on computer tape. In 2024, the space probe is still collecting and sending valuable data back to the earth with the same old hardware. While extreme technological developments have gone by on earth, the old hardware from 1977 still needs to interact with the probe. Furthermore, the metadata, for example, of the sensors built into that probe, must also be preserved to extract information from that data. [HP12]

Furthermore, Space missions generate massive amounts of data over many years, which must be stored for decades, if not forever, to support historical research as well as future scientific investigations or analysis. Long-term mission data integrity, accessibility, and usability pose logistical and technological challenges.

3. Related Work

In the following section, similar flight data storage systems will be introduced to get an idea of what has already been done. In addition,various guidlines for long-term preservation of data are presented.

3.1. Similar Databases

Two archives that follow a similar approach to the for this thesis developed archive are presented in the following.

3.1.1. ESA Planetary Science Archive (PSA)

Starting with the Planetary Science Archive of ESA. NASA was the first space agency to discuss a scientific archive and created the Pilot Planetary Data System (PPDS) in 1984 and later the Planetary Data System (PDS). In 2000, after ESA's first planetary exploration systems, they developed the PSA, which should become the main storage and archive point for all coming scientific missions by ESA. It has become one of the mature entities in archiving data from space missions. [BVB+18]

They committed to preserving the data sets they produced for decades or even longer. To ensure this commitment, they are implementing the IPDA standard to help the archive be available, readable, and usable in the long term. The IPDA defines a standard format called PDS3 format, which defines next to how the data should be stored also the metadata that has to be stored to make it possible in future decades to understand how the data was obtained and how they should be read and interpreted. In 2016, the PDS3 format was updated to represent new technological developments, and now the PSD4 format is the standard for data archives that implement the IPDA. [BVB+18]

In general, the archive is divided into two different parts. The first one is the actual data archive, where all the data aligns with the regulations for long-term storage. The other part is the operational archive. There, the data currently used gets ingested. The data in the operational database has to go under various reviews before getting stored in the archive. Next to the compliance with the PDS standards, they also get checked by actual scientists, who are asked three fundamental questions:

- Will the scientific products mislead the scientists in their scientific interpretations?
- Does the user have all the necessary information to produce scientific results?
- Is it possible to reproduce science results published by the instrument science team without additional software or information to access and read the data?

This review can be compared to a scientific peer review, and after the process, the data is scientifically checked and ingested into the archive. To access the data, the PSA provides certain tools like a user interface.. [BVB+18]

3.1.2. German Satellite Data Archive at DLR

The German Satellite Data Archive (D-SDA), located at the German remote data sensing center of the German aerospace center DLR, is a data management system providing archiving and access functionality for earth observation data. It has operated for over two decades and has collected about 3.8 petabytes of earth observation data. Allowing a throughput of around six terabytes per day and a total capacity of 50 petabytes. [KMS+14]

A key goal of the archive is to provide the collected data for future generations. In the achieve, a unique snapshot of the earth and atmosphere at a specific point in time is stored. This data can be very valuable for research in the near and far future to better understand how the earth and its climate are developing over time and maybe find new insights that would not be possible without that data. The scientists at the DLR are aware that the preservation of data has two sides. The hardware side, to ensure that no data is lost because of hardware or software issues, earlier called bit preservation and the preservation of the information stored in data, the logical preservation. [KMS+14].

To conclude, extensive data management in earth observation is not just a sole IT problem but also one that requires interdisciplinary collaboration. A challenge involves developing processes, maintaining high-quality metadata, and ensuring ongoing accessibility and usability of the data. By following guidelines and best practices, organizations like DLR aim to guarantee that the data remains accessible, reliable, and meaningful for future generations, contributing to a deeper understanding of our planet.

3.2. Guidelines for Data Preservation

Next to archives that are in use already, different organizations developed,as mentioned before, various guidelines for data preservation. These present policies and approaches to achieve preservation and long-term storage of data. In this context, long-term means at least a decade, but optimally for as long as the data has any value. Much work was put into these guidelines, and it often took decades to develop the current version, with updates and iterations to adapt to the evolution of technology. In this section, different guidelines from different companies are presented.

3.2.1. The International Planetary Data Alliance

The awareness that data storage and archiving are very important, not only for documentation but also for all future space missions for error search or similar, has increased over time. As outlined in the introduction, it only became apparent after the first space missions, and from there on, much work was put into building such archives. Furthermore, various space agencies had ambitions to build archives together to gain even more insights from the data. Nevertheless, there were problems in the form of data types and procedure

differences. Also, the interests of different space agencies might be different (archives only for the scientific community, for the public, or only for their country). All share the same interest, which is the preservation of data for an extended period of time. This led to the development of the nternational Planetary Data Alliance (IPDA) by NASA and ESA. [BVB+18]

The IPDA, a global initiative founded in 2006, is dedicated to developing standards for archiving data from scientific missions in the exploration of the solar system. It is a collaborative effort that initially involved agencies in the United States and Europe, such as NASA and ESA, and has since expanded to include other agencies worldwide, including Japan and India. This global collaboration underscores the shared commitment to advancing planetary exploration through data archiving standards.

To achieve this, they defined standard data types and a reference architecture so that different agencies with different archives can work seamlessly together. This reference architecture shows the processes and architecture for planetary archives to collect and share data. In detail, it consists of the reference architecture of three different sub-architectures: [COM+09]

The Process architecture defines standard processes for planetary archive systems like archive management, preservation planning, and peer review. The data architecture refers to the organization and design of an entity's data assets, both logical and physical, as well as the resources used for managing this data. Key components are standard data formats and modeling objects and their relations. The third sub-architecture is the technology architecture, which defines a set of standards and components to allow interoperability between different planetary archives. [COM+09]

In 2008, the Committee on Space Research (COSPAR) chose the IPDA to set the standards for planetary archives. Since then, it has led international cooperation in archiving scientific mission data of planetary exploration and long-term archiving of this mission data. In contrast to the two guidelines in the following sections, these recommendations concentrate on building archives compatible with the existing archives of NASA and ESA. And especially for planetary exploration missions. Both topics are not as relevant for the archive described in this thesis, and therefore, the IPDA is not presented in detail. [COM+09]

3.2.2. Reference Model for an Open Archival Information System (OAIS)

The Reference Model for an Open Archival Information System (OAIS) works as a framework for organizations intending to build and manage an archival system for long-term preservation. It was developed by the Consultative Committee for Space Data Systems (CCSDS). It is defined as a system responsible for preserving information for long-term access and providing the capability to access the preserved information. [SRMP02]

The OAIS consists of three different participant groups. The producer is the entity that submits the data to the archive. The consumer, the data's end-user, accesses the archived information, and the management monitors the OAIS system and ensures it meets the organizational goals. The core concept is the preservation of information objects, which consist of data and the representative information needed to understand the data. Thereby are different functional entities defined. The ingesting process is responsible for receiving and

processing information from producers. Archival storage handles the storage, preservation, and management of archival data. The data management entity maintains the metadata, and the administration manages the system's operations and policies. The Preservation planning entity addresses technological changes and obsolescence to ensure the long-term usability of the archived information. Furthermore, the access ensures the user can access the archived data and services. [SRMP02]

The reference model divides the data packages into three different kinds, depending on the state in which the data is. The first one is the Submission Information Package (SIP), which describes the package sent by the producer. The second package is the Archival Information Package (AIP), which consists of the data and the metadata and is the package stored in the archive. The third package is the Dissemination Information package (DIP), provided to the consumers upon request. An overview of the system can be seen in Figure 3.1, where the different interactions between entities and participants with the according packages can be seen. [SRMP02]

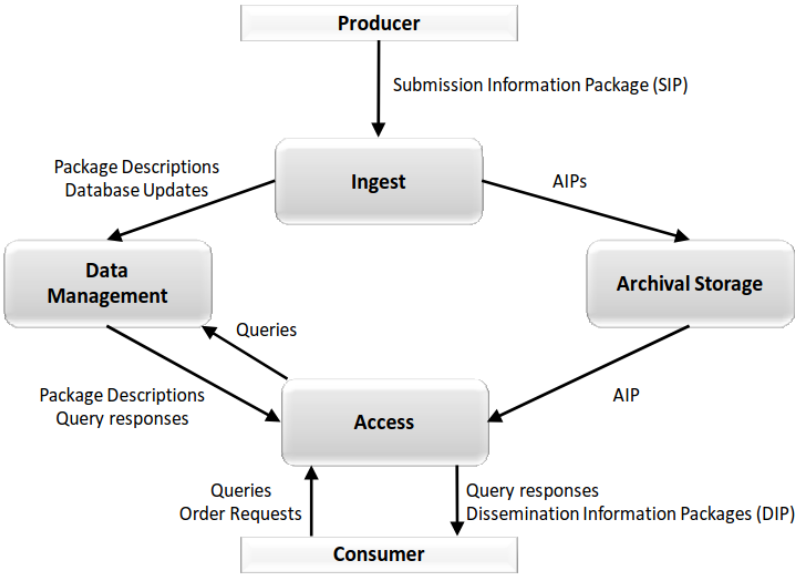


Figure 3.1.: High-Level Data Flows in an OAIS [SRMP02]

A key aspect mentioned by the OAIS is the importance of preservation metadata, including technical, descriptive, and administrative information, to ensure the data can be accessible and understandable in the future. However, it is important to note that these goals should be focused on a designated community for which the archive is built. [SRMP02]

3.2.3. Revised GEO Data Sharing and Data Management Principles

The Group on Earth Observations (GEO) is a global network connecting research, academic, and government institutions to create innovative solutions for challenges in all areas of earth observation, including climate change, biodiversity loss, and pollution [Wor24].

In 2015, the GEO data management principles task force defined a common set of GEO data management principles. These principles should cover five topic areas: discovery, accessibility,

usability, preservation, and curation. Each topic is defined by one or more Data Management Principles (DMP). These principles can act as a guideline for all earth observation products, both remote sensing and in-situ data, and cover raw data as well as high-level products, including analysis-ready and on-demand produced data products. [GEO24]

Discovery

The first category is Discovery with only one DMP describing how the data and the according metadata should be discoverable through catalogs and search engines. Outlining specific metadata necessary to discover the required data, including title, description, geographic location, identity, temporal coverage, keywords, conditions, and restrictions. Furthermore, it is stated that metadata should be checked periodically for validity and correctness, especially for cross-reference links to related data. Creating a metadata set for each data set is a labor-intensive activity. [GEO24] It seems very expensive in the short term but is profitable in the long term. An example from the Dutch Digital Bewaring project cited by Beagerie et al. [PETS12] shows that creating a batch of 1000 records with well-written metadata costs 30 times less than the same 1000 records with badly created metadata.

Acessibility

The accessibility topic describes how users of the data archive should have access to the data. It also consists of one DMP. In detail, the archive should provide an online interface where the data is available on-demand, and the user can find and download the required data. It is preferable to provide a user-customizable service to access, visualize, and analyze the data. Therefore, data processing should be done in place, and the service architecture should be simple using tools like jupyter notebooks. Moreover, a user management system should be implemented to handle different user groups. For the various services, vendor-lock-ins should be avoided to ensure the archive's future and avoid unnecessary migrations. [GEO24]

Usability

The usability topic consists of four different DMPs ensuring the usability of the data archive. The first DMP concerns the encoding of the data. The GEO data working group recommends using widely accepted encoding, targeting primarily the user community. Additionally, international standards are preferable because then various data transformations are unnecessary, and the interoperability is simplified. The next DMP outlines the necessity of data documentation through metadata. In particular, to document all necessary elements to access, use, understand, and process the data. If possible and practicable, the metadata collection should follow international or community-approved structural standards. Important metadata often forgotten are variable meanings and units, and overall, various kinds of metadata should be collected, including descriptive, structural, and administrative metadata. It should be the responsibility of the producer and distributor to create the metadata because costs can be reduced dramatically if the metadata is produced by the same people at the same time as the related data is produced. [GEO24]

Data Traceability is covered by DMP-5, describing that the metadata should also include the origin and processing history of the raw data. This includes processing algorithm

descriptions, the provenance, the transformations made, and the involved parties. This ensures that the data and its production chain are fully traceable and leads to a better understanding and comprehensibility of the data. If possible, an automated creation of this metadata would be preferable.
The last DMP of the Usability aspect recommends that there should be a process to control the quality of the data and the metadata. There should be some indication that the quality control was correctly performed, and data that is not yet quality controlled should be flagged. Generally, it should be checked at least two times, firstly by the data producer and then by another entity. [GEO24]

Preservation

The next aspect of data that is covered is preservation. In this case, the bit preservation as described in Section 2.6.1. The first of the two principles describes activities and processes needed to ensure data preservation. This includes planning the preservation, meaning scheduled transformations, system backups, and recovery plans. This includes multiple processes that should be implemented. The first one is archived data refreshment. The process describes that the data should periodically be migrated to the currently most adequate proven technology. The technology selection should be based on technical, cost, and environmental aspects. The next process is duplicating the archived data, meaning maintaining two identical copies of the archive. Thereby, it is differentiated between four different security levels: The first is having two copies in the same geographical location to avoid data loss due to media degradation. The second security level is to have two copies of different technologies in the same geographical location to avoid technological-based failures. The next level is to have two copies in two different locations to prevent data loss through external factors like floods. The highest security level is to have two copies in two different geographical locations on two different storage technologies. The last process in this DMP is the hardware migration of the archive system components. Not only the data stored in the archive should be updated, but also the hardware of the remaining system components should be updated periodically. [GEO24]

To ensure long-term preservation for different file formats, formats that contain data loss through lossy compressions should be avoided. The format TIFF is presented as an example of a format to store images in a lossless format and also includes metadata of the file in header tags. [Mas23] [GEO24]

The second principle is the data and metadata verification principle. It describes that all the stored data and metadata should be periodically verified to ensure integrity, authenticity, and readability. This includes tests to check the readability and accessibility of media, as well as verification of the archive and data integrity.

Curation

The last topic area of the GEO data sharing and management principles is data curation. The first principle focuses on correcting, updating, and reprocessing the data. Updates and corrections are needed and recommended as new technologies evolve, or the data gets

reviewed by different user groups to have an as correct data archive as possible. In some use cases, reprocessing can lead to higher-quality data. It is also often necessary after updated instrument calibrations, sensor degradation, or new insights in the data. The last principle covers the persistent and resolvable identification of the data. This is useful for any kind of communication about the data. It also includes the possibility of citing the data and giving acknowledgments to the data providers if necessary.

3.2.4. European LTDP Common Guidelines

To address the challenges of a long-term data archive for scientific research, in 2007, ESA founded a workshop group to build long-term data preservation guidelines for earth observation. So, to show how to store scientific data for the long term and what is needed to achieve it. This resulted in 2010 in the document 'European LTDP Common Guidelines' [ABD+10]. This document defines an approach and guidelines on how to store earth observation data for a long time. The Data Stewardship Interest Group (DSIG) of the Working Group on Information Systems and Services (WGISS) from the Committee on Earth Observation Satellites (CEOS) then used this document to build a global reference for earth observation data preservation. It published the document "Long Term Preservation of Earth Observation Space Data Preservation Guidelines" in 2015, with an update in 2023 [AMMC23]. This document not only defines an archive but also ensures and facilitates the accessibility and usability of the preserved data.
This document defines eight guiding principles and themes presented in the following:

1. Preserved Data Set Composition
2. Archive Operations and Organization
3. Archive Security
4. Data Ingestion
5. Archive and Data Maintenance
6. Data Access and Interoperability
7. Data Exploration and re-processing
8. Data Purge prevention

These key themes define the principles that have to be covered, when a long term preservation archive is build with a focus on earth observation. [AMMC23]

Preserved Data Set Composition

The guiding principle of this theme is that the archived data should contain everything necessary to be accessed, used, understood, and processed in the future [AMMC23]. This was presented in Section 2.6.1 as logical preservation, so to define what is needed to ensure the data is usable in coming years. Thereby, preserved data is divided into three different kinds of data. The first one is the raw data of the missions, like sensor data, but it also includes metadata like calibration values, etc. The second type of data is data about the processing software used during pre- and post-mission data generation. This should be

stored to make it reproducible how certain data was generated. The third kind of data is everything that comes with mission documentation, including architecture, algorithms, procedure descriptions, and performance reports. These three types of data together form a data set content that allows preserved data to be usable and understandable over a long period of time. It is important to connect these three types of data in the archive to form a complete data set for each mission. [AMMC23]

Additionally, the formats of the data, as well as the export and documentation formats, have to be defined to ensure future accessibility. Standard data formats should be used to avoid incompatibility issues due to technological obsolescence. Thereby, these formats should be sustainable and lossless ensuring data integrity like TIFF or SAFE. [AMMC23]

Archive Operations and Organization

Archive operations and organization include all daily activities that have to be done to monitor the archive system. The guiding principle is to automate everything to a maximum extent, and the operations should only be performed by trained personnel following specific, well-documented procedures. This includes keeping the archive equipment within the manufacturer's recommendations. Additionally, specific persons should be responsible for maintaining and observing the archive. [AMMC23]

Archive Security

The archive security includes all activities to guarantee the archived data's confidentiality, integrity, and availability. This includes security measures that only authorized persons have physical and virtual access to the archive. Furthermore, certain precautions must be taken to protect the archive against external factors like floods or fire. [AMMC23]

Additionally to unauthorized access, there also have to be mechanisms that prevent authorized users of the archive from seeing confidential data because not every data is available to all users. At last, it includes measurements to work against unintentional or deliberate human actions or technical imperfections that can result in data loss. [AMMC23]

Data Ingestion

The data ingestion topic defines how the data gets into the archive. This means this is the connection point between the data producers and the archive. Therefore, interfaces have to be defined to ensure the data sets' compatibility and the archive's usability. [AMMC23]

To archive this, an ingestion process has to be defined, where the user generates or adds the needed metadata. Then, the preserved dataset has to be adapted to the archive standard defined in Section 3.2.4 to make the data understandable and sustainable. After that, the data should be checked (automatically and by the archivist) to ensure all the standards are met. Another suggestion is to document all ingested operations and to have an overview of what is stored in the archive. [AMMC23]

Archive and Data Maintenance

The archive maintenance guideline principle suggests that the archive holders should design a maintenance mechanism that ensures the integrity of the archived data, which means that the stored data is complete and unaltered. This mechanism is mostly a set of performed routines for hardware exchange and migrations to new systems and media according to technological evolution. Part of this maintenance operation is to ensure the bit preservation, explained in Section 2.6.1. [AMMC23]

The first step would be to periodically migrate the achieved data to the current most suitable technology. A typical period to do this is every five years. This ensures that the technological basis is not outdated and that data access is preserved. Together with the technology refreshment, the standards used in the current system should also be revised. This means it should be evaluated if the standard formats and exchange formats have changed, and if so, migration and reformatting to these standards should be done. It makes sense to do this with migrating to a new technology. [AMMC23]

Another mechanism to ensure the completeness of the data and to protect the archive against loss is data duplication. Thereby, different security levels can be implanted. From copies in different locations to copies on different media. On top of the periodic technology and format updates, the archive should have a test life cycle to ensure everything is working as intended or modifications have to be made. [AMMC23]

Data Access and Interoperability

The goal of an archive is for data to be stored and usable for analysis or other insights over the long term. Therefore, the archive should provide an interface or a way to access the data. This system should provide standard and common access to the data despite the differences in the different data sets. Thereby, there should be a mechanism to access the raw data and the metadata. Of course, the way of accessing which kind of data is highly use-case dependent and, therefore, has to be specified for each application. However, there should be different access points based on the three types of data defined in the theme about data preservation set in Section 3.2.4. [AMMC23]

Part II.

Problem Statement and Solution

4. Problem Statement

In this chapter, MORABA, the department of the DLR with which this thesis was made in collaboration, will be briefly presented alongside the current state of data management and this department and the requirements for the flight data archive.

4.1. MORABA

MORABA is a department of the institute Space Operations and Astronaut Training of the German Aerospace Center (DLR). While based in Oberpfaffenhofen, the name MORABA translates to Mobile Rocket Base, and they developed a mobile infrastructure to launch a rocket from anywhere on Earth in a short period of time. From planning to preparation and implementation to the launch and post-flight analysis.

MORABA is specialized in providing launch and support services for scientific experiments using sounding rockets and balloons. These experiments cover various fields, including atmospheric studies, microgravity experiments, and space science. MORABA launches five to 15 rockets annually for various use cases. The department is also involved in various research projects, such as reusable rockets and hypersonic research, as well as international projects.

MORABA was founded in 1967 and has carried out almost 500 campaigns worldwide since then. Reaching from Australia, USA, Brazil to Greenland and Norway. Next to NASA, it is the only institution in the world that can perform scientific high-altitude missions from everywhere in the world.

4.2. Requirements of the System

When starting an extensive number of rockets, a lot of data is produced naturally. There are many different sensors on the rocket and on the ground that monitor it. These massive amounts of data are very valuable because they are only produced once during the flight, and it is very hard or, in most cases, impossible to reproduce this data. Therefore, it makes sense to preserve this data. In space exploration and rocket science, missions often last several years to decades. With research and experiments, failure naturally comes at certain points, and it is very valuable to have the possibility to look into older data of past flights or missions to find a solution or at least understand the problem. Therefore, it makes sense to preserve this data over a long time. To achieve this and understand older data even after decades, good data management is essential.

Currently, at MORABA, the data management and storage do not follow a unified guideline or policy. There is also no central point where all data collected from a mission is stored. Rather, there are different tools to store the respective data. MORABA is divided into different subgroups, from a group handling the sensors and telemetry to a group handling the produced data. Every group has its way of storing the data it needs or produces. If they need data from other groups, they communicate directly with the members of that group to get the needed data.
This produces the problem that the employees always have to find the data they need and that there is no central system where the data is stored. Without a certain process and architecture to store data and without a long time preservation process, using data from 10 years ago might be possible. However, everything older than that is not manageable. Firstly, because the raw data might not be available anymore, and secondly, even if it is available, important metadata like calibration values or what kind of sensor was used is most likely lost.

So, the main problem is to build a flight data archive, where mission data is stored for long-term preservation, and the data will be usable and interpretable in the future. There are a couple of high-level requirements for such a system. The first one is that it has to implement certain functionalities and processes to ensure data preservation over a long time and that the data stays available and understandable. Furthermore, it has to include certain security mechanisms that enable the hiding, for example, of confidential data from users who do not have access rights. Additionally, interfaces and possibilities have to be defined to allow to fill the archive with data as well as for users of the system to export the data.

5. Methodology

The requirements of a flight data archive are highly dependent on the use case and the user. To find the requirements for such a system, much work must be done to understand the problem and what is needed. In this case, the requirements and the solution were developed in close collaboration with MORABA. As mentioned, MORABA is divided into different subgroups with different tasks, but all depend heavily on each other. Additionally, they all work on the same data sets for the same missions. Different aspects of the data have different importance to the subgroups. Each group is involved in different work with the data. One is responsible for processing data, and another does a lot of analysis. So because the system is built for all employees at MORABA, to find the requirements of the system, every group has to be considered.

To start and get an overview of the problem, at least one member of every group was consulted to gather their opinion on the matter and their specific needs in the particular group. Key requirements were identified during these conversations, and a base concept was developed. This includes what data should be stored in the flight data archive. This is

- all kinds of sensor data and measurement data like weather, GPS, housekeeping data etc.)
- Metadata and context data complementing the measurement data
- images and videos of the flights
- configuration data
- postflight (protocol) documents

Next to what kind of data should be preserved, key requirements were defined:

- easy and fast accessibility
- possibility to extract data in different formats
- storage of raw data as well as processed data
- storage of processing pipeline for traceability of data processing process
- flexibility to consider differences in flights (architectures, sensors etc.)
- possibility to store, next to flight data, also data of experiments and tests prior to the launch

After the development of the base concept, it was reviewed with the scientists at MORABA to ensure accuracy and relevance. Then, a literature review of the subjects related to the topic was made. Using insights gathered from both the literature and the discussions with scientists, the proposed solution was developed. During the development process, close communication with the scientists was maintained to ensure the goals were met.

The key challenges were finding a storage solution that is flexible enough to accommodate the complexity of high altitude and space flights, making the archive accessible and discoverable, and finding all meta and context data necessary to ensure the long-term preservation and usability of the data.

6. Solution

The challenge of long-term preservation of MORABAs space missions can be divided into two main categories. The first focuses on developing requirements and guidelines to ensure the data remains interpretable and usable for many years. This involves identifying what types of data need to be preserved and determining the necessary metadata and additional information to retain. This ensures that as little information as possible is lost over time, allowing future comparisons and analyses. Next to this, the data's physical preservation has to be ensured.

The second category is to develop and present a practical platform that meets all requirements to store the data, which can be derived from the guidelines and policies developed in the first category. So, to present a reference architecture capable of storing all the required data and information, making it accessible and usable.

However, in the end, these two categories are two different sides of the same problem. One more practical and one more theoretical. The idea of this section is to explain how long-term time data preservation can be achieved, specialized for space flights, and to show, with the help of a prototype, how these concepts can be implemented.

6.1. Characteristics of a Flight Data Archive for High Altitude and Space Missions

However, before describing the theoretical and technical aspects of the provided solution, it is important to explain the core characteristics of a flight data archive for sounding rocket missions with a special focus on the MORABAs use case. This helps to understand how to fulfill the requirements presented in Section 4.2.
To identify the key characteristics, the different concepts presented in Chapter 2 about the Theoretical and Technical Backgrounds are specified for the archive in the following.

6.1.1. Storage Concept

In Section 2.1, various concepts of databases were presented. In the following subsection, a closer look at these concepts is taken to examine with which a data archive aligns.

Hot versus Cold Storage

Starting with the difference between hot and cold storage solutions (See Section 2.1.1). An archive is more of a cold storage system, which gets accessed less than a system like an operational database, which is involved with data currently worked on. However, in this

case, the data still has to be available on demand, because a flight data archive has, next to preservation functionality, also operational functionality.

While not having data of ongoing missions stored in the archive, the content of the archive often has value for ongoing flights and missions. Therefore, all the data stored inside has to be available on demand. However, when designed for a company with around 60 employees, simultaneous access is rather rare compared to, for example, a publicly available archive.

The key factors differentiating between a hot and a cold archive, as described in Section 2.1.1, are access frequency and retrieval speed. The flight data archive for MORABA will have a moderate access frequency, and the retrieval speed has to be high enough not to disturb the ongoing workflow but also does not have to be instantaneous for automatic processing or something similar. To summarise, the flight data archive is a hot storage solution with relatively low performance requirements.

Cloud versus On-Premise

The next question to consider is whether the archive should be operated with the help of a cloud provider or hosted on-premise. As outlined in Section 2.1.2, this is a question of performance and data security. For scientific high-altitude and space flights, rockets or boosters are used. These often have military backgrounds. Therefore, at least part of the flight data is confidential and must be protected against unauthorized access. Therefore, hosting the archive on a third-party managed cloud system is almost impossible because knowing who can access the data is difficult. Furthermore, the flexibility of cloud storage solutions is unnecessary for this application because with 60 users that can access the archive, up or down scaling does not provide any substantial advantages.

Static versus Active Storage

The third and last concept discussed for storage solutions was active versus static storage. The idea of the flight archive is that data gets only ingested once after the related mission or flight is over. This implies that all the data stored in the archive is "correct" and will not be changed again. As described in Section 2.1.3, the active modification is the distinguishing factor between an active and a static archive. Therefore, the archive is a static archive.

So, to summarise, based on the requirements and nature of the data, a flight data archive is a static hot archive stored on-premise inside the company.

6.1.2. Data

This section discusses the different aspects of data inside the flight archive. This is important for choosing a suitable database and is the heart of the archive. As described in Section 2.2, the different aspects of data can be described through the five V's: Variety, Volume, Velocity, Value, and Veracity.

Data Variety

Starting with the data variety. When looking at examples of flight data from MORABA, three different types or areas of data can be specified. The first is the flight data from the various sensors onboard the rocket and the ground station. This is mainly structured data with a timestamp. Then, there is the metadata and description data. Most of this data is key-value data and can be categorized as semi-structured data. This data describes the flight data and the flight in general, like the rocket's geographical launch location or what engine was used. Also, the data needed for long-term preservation and discoverability of data is part of this type of data. The third kind of data is unstructured data, such as images and videos of flight or post-flight reports.

This shows the complexity and variety that a flight data archive has to preserve. It consists of all three common types of data presented in Section 2.2.1. In Figure 6.1, the different data types can be seen, with examples of which specific data they are representing.

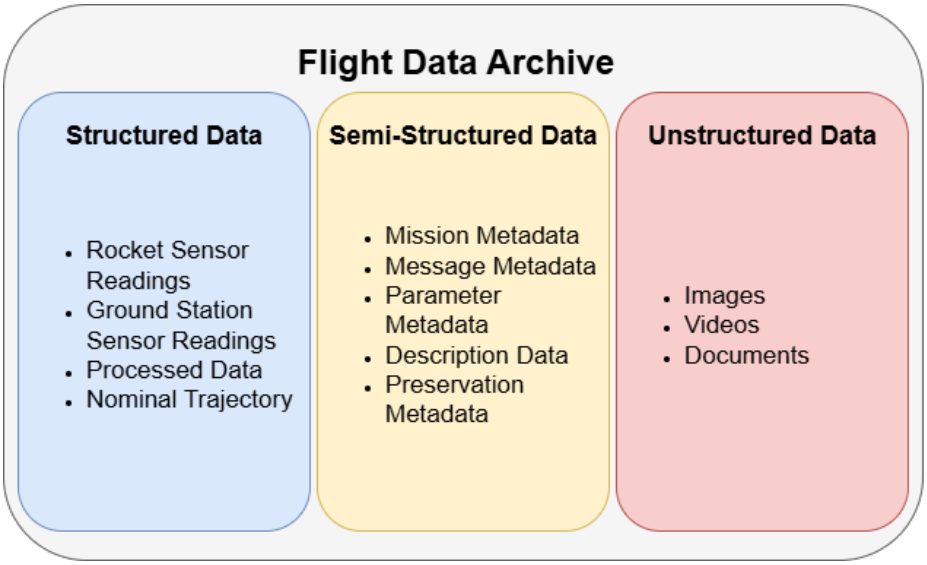


Figure 6.1.: Content of a Flight Data Archive

Data Volume

The next data concept is data volume. A rough estimation of the data volume of the raw data has been made in Section 8.4. However, the volume of the data varies a lot between the different missions. It highly depends on how many sensors are on the rocket and, for example, how many cameras capture the flight. Therefore, it is not easy to estimate the data volume accurately. However, considering the general trends of increasing data sizes discussed in Section 2.2.2, it is clear that the archive must be designed to accommodate high future growth in data volume.

Data Velocity

The third concept is data velocity, which describes the rate at which data is generated and processed (see Section 2.2.3). This storage solution should act as a data archive. Therefore, the data will be uploaded to the archive as a bulge after a particular mission or flight is finished. So, there will be no immediate processing or analysis of data as it is being uploaded. Consequently, the archive does not need to accommodate continuous or instantaneous data ingestion. Furthermore, modifications to the stored data will be infrequent. Therefore, the performance criteria are not as strong.

Data Value

The fourth concept is data value, which describes the importance and utility of the data. Rocket flights are very complex and expensive. Therefore, scientific rocket missions are often unique opportunities to collect data. Reproducing the data is very expensive at most, rather impossible. Therefore, almost all the data collected is very valuable and should be stored in the archive. Also, because space missions often last for multiple years, the collected data stays valuable longer than in other faster-paced experiments or operations, just because the frequency of newly produced data is relatively low. In Section 2.2.4, it is described that only 2% of generated data is stored longer than one year. This will be considerably higher for rocket missions because of the reasons mentioned earlier, where only very little unnecessary data gets generated.

Data Veracity

The fifth and last concept is data veracity, which describes the quality and reliability of data (see Section 2.2.5). For rocket flights, data veracity is essential. It ensures that the collected data is accurate, reliable, and trustworthy. This is crucial for making critical decisions, and because there are few opportunities to collect the data, it is essential that the collected data is reliable. To ensure high data veracity, many mechanisms are implemented before the data gets stored in the archive. This includes sensor calibrations, redundancy, and cross-verification for data like location information.

6.2. Long Term Data Preservation of MORABAs Mission Data

In Section 2.6.1, the general background of long-term data preservation has been described, and in Section 3.2, multiple guidelines for various applications of data preservation with a focus on space have been presented, including physical and theoretical requirements. This section will cover and adapt the different aspects of data preservation to the unique requirements of high altitude and space flights, with a particular focus on MORABAs use case, based on the data characteristics in the previous section. In the end, this section forms a scientific basis for building a flight data archive for sounding rocket missions. This should ensure the four core functionalities a data archive has to fulfill: Usability, Accessibility, Discoverability, and Preservation.

The structure of the following section is oriented at the structure of the European LTDP common guidelines for earth observation presented in Section 3.2.4 from [AMMC23]. These

provide a complete overview of the requirements for long-time data preservation in earth observation and can be adapted and specialized for this use case. While there are key differences between earth observation and short-duration spaceflights – such as mission objectives, types of data collected, and operational challenges, the core principles of data integrity, security, accessibility, and preservation hold for both fields. The established practices in earth observation data preservation offer a tested and reliable framework that can be adapted and specialized to accommodate the aspects of space flights. Additional information from the remaining guidelines presented in Section 3.2 is also used in the following. This ensures that the following scientific basis is built on a robust foundation.

6.2.1. Preserved Data Set Composition

The preserved data set composition defines the data set that should be stored. The goal is that this data set ensures that the data can be accessed, used, understood, and processed in the future. As for earth observation, the data set for space flight can be divided into three categories. This is oriented heavily on the nature of the data and correlates with Section 6.1.2 and Figure 6.1.

Measurement Data

The first category is the measurement data, which is the core information that is stored. This includes all data that is collected during the flight as well as all data that is produced during pre- and postflight processes. This can also include data from tests and experiments or calculated data like a nominal flight trajectory of the rocket. Compared to earth observation data, the data from rocket launches is produced at a much higher frequency. So, there is a lot more data in a shorter period of time. However, the duration of a flight is also much shorter than, for example, an earth observation satellite.

Context and Metadata

The second category is the context and metadata supporting the raw data. This is all data needed to make the data interpretable and analyzable. This data should be collected during or shortly after the mission. Furthermore, this data can be used to make the data better discoverable. This follows the concept of the OAIS presented in Section 3.2.2. The data that is necessary and should be stored depends on the use cases and should be developed together with the data users. Defining a standard set of context data collected for each mission is crucial. This helps to ensure that all necessary data is available for each mission and ensures comparability between different missions.

The metadata set for MORABA is structured as close as possible to the nature of the data. This means each mission can store meta- and context data at three different abstraction levels. The first one is data that is related to the whole mission. This includes data like launch time, launch location, and launcher settings. The second level is the "message" level. A message usually describes a device or sub-device that sends a set of related parameters. So metadata like device version or similar information can be stored at that

level. The lowest level is the parameter level. A parameter is part of a message, which is part of a mission. The information specific to the parameter gets stored there. This is partly information to enhance discoverability like a parameter category, but also information specific to that parameter. In Figure 6.2, a visual representation of this structure can be seen.

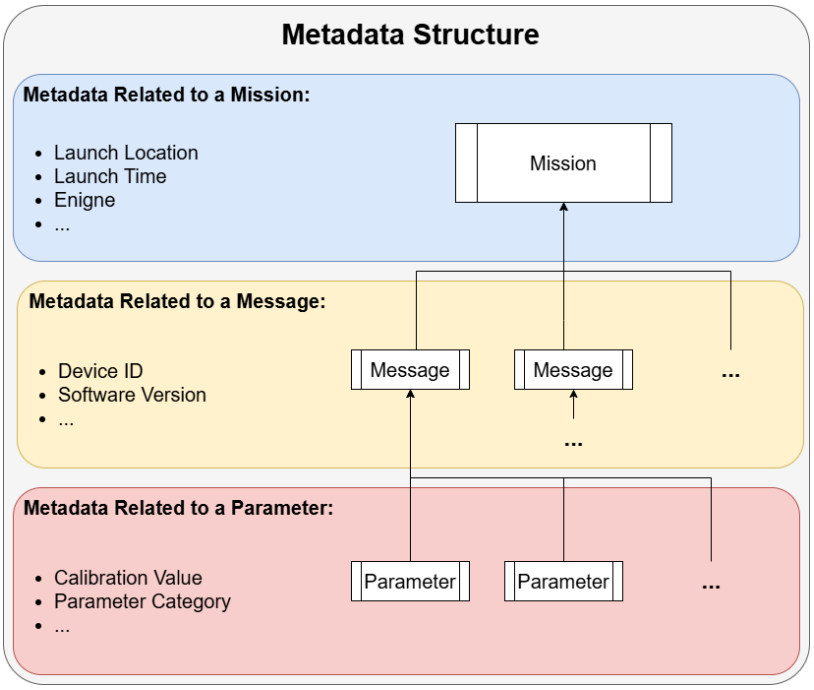


Figure 6.2.: The Metadata Structure of the Data

Unstructured Data

The third category, completing a data set, is unstructured data like images, videos, or documents. It is advisable to use standardized data formats already used and widely accepted in the user community. In this case, this applies primarily to unstructured data like images, videos, and documents because the rest of the data is stored in a database with its own formats. As described by the GEO data sharing and data management principles as well as by the LTDP common guidelines in Section 3.2, these formats should ensure data integrity by using sustainable and lossless file formats. In Section 2.6.2 this is described in more detail. [GEO24] [AMMC23]

6.2.2. Archive Operations and Organization

This subsection describes all procedures and operations that must be done to maintain and operate the archive daily. This includes maintaining the update life cycle of hard- and software and being responsible for solving problems that might emerge. The idea is that

at least one person takes over the role of an *Archivist*. This role can be compared to the management role in the Open Archival Information System (OAIS) [SRMP02].

This person is then responsible for the mentioned tasks. In this way, it is ensured that a knowledgeable entity manages the archive that is familiar with it, and therefore, it minimizes the risk of errors. Designating a dedicated person ensures the integrity and consistency of the archived data. Having a central point of accountability facilitates troubleshooting, continuous improvement, and adherence to the established data preservation standards.

6.2.3. Archive Security

The security of the archive can be divided into two aspects. The first one is authentication and access control. The archive has to ensure that only authorized persons are allowed to gain access to the data, that they are allowed to access. In MORABAs case, this will not be a public archive. Therefore, no public access control has to be implemented. This will most likely be for all flight data archives of rocket launches because often, the used rockets have a military background and are subject to confidentiality requirements. Nevertheless, this does not imply that there are no internal differences in users' access rights. However, it is common for organizations to already have mechanisms in place that define and regulate who is permitted to view specific data sets.

Therefore, the archive does not need to support a full, standalone user management system but rather to adapt and use the existing mechanisms. This separation of concern is a requirement defined by Mohammed et al, presented in Section 2.5. In this way, the system can ensure compliance with established policies and procedures while avoiding redundancy and the complexities of creating a parallel management system. The system should be able to integrate the existing mechanisms and apply them to the stored data.

In the context of space flight and rocket launches, a very sophisticated mechanism might be needed. In the example of MORABA, some parameters like acceleration are confidential during the launch because, with the help of these, conclusions on engine characteristics are possible. However, this is no longer possible once the rocket is in space because the engine is shut down. Therefore, these parameters should be made available, for example, for the scientists conducting experiments on the rocket's payload to eliminate external influences. So, access control to the level of parameters and timestamps has to be implemented, which is considerably more complex than one at the level of missions, messages, or even parameters. This means there has to be the possibility of filtering the data based on different attributes, like timestamps, based on the current user role. This follows the concept of varying granularity of protected data objects described in Section 2.5 about security, so the implementation of access control on different levels of the data schema.

The second aspect of archive security is protecting against data loss for any reason. This aspect is covered in Section 6.2.5 about archive and data maintenance.

6.2.4. Data Ingestion

Data ingestion defines how the data gets into the archive. As described in Section 3.2.4, interfaces should be defined to ingest the data and the related metadata at the same time. Automatic data creation is not as important for a flight data archive in space flight because, as stated in Section 6.1.2, the data comes in batches and is not live during a mission. However, as much automation as possible is still advisable because it is less error-prone and less labor-intensive than when entered by a human.

During the ingestion process, multiple different tasks have to be fulfilled to ensure proper integration into the archive. The measurement data has to be transformed into the schema of the data archive to maintain continuity across all stored data. Additionally, it is essential to establish a mechanism for defining detailed access rights to the data. As discussed in the previous section on archive security, access control must be implemented at a granular level, allowing permissions to be specified for individual parameters within defined time ranges.

Furthermore, the metadata and context data for the specified standard data set must be collected. This process should combine as much automation as possible with selective manual input. The interface should automatically identify whether parameters or messages are already associated with previous flights when, for example, a device and sensor are part of multiple flights. When such matches are detected, the system should provide the stored information to the user, enabling them to decide whether to retain, modify, or add new information.

The metadata collection is particularly complex in the context of rocket launches because each rocket is uniquely configured, featuring a unique combination of devices and instrumentation. Therefore, the data generated by each flight is also unique. In contrast, satellite missions for Earth observation involve satellite structures that remain consistent throughout their longer operational lifetimes. This consistency reduces the frequency of changes to the associated metadata. However, with rockets, the variability in structure and shorter mission duration lead to more frequent updates to the metadata set, significantly increasing the complexity of managing and maintaining it.

Therefore, it makes even more sense to implement an ingestion cycle, where a second entity checks the data and metadata for correctness as stated in the European LTDP common guidelines presented in Section 3.2.4 and also implemented by ESAs PSA, described in Section 3.1.1. This should be the task of the in Section 6.2.2 introduced *archivist*. In this way, two different entities are involved in the data ingestion process: the first one, optimally the data producer, who has a lot of knowledge of the data, and the second one, the archivist, who has a lot of knowledge about the archive. This combination forms a reasonable basis to ensure the integrity and correctness of the archive. In Figure 6.3, the ingestion process and the according tasks done by the data producer and the archivist are shown.

6.2.5. Archive and Data Maintenance

The archive and data maintenance procedures can be adapted similarly to those from the earth observation archives. The maintenance routines to update and maintain hardware and software can also be applied to a flight data archive for space flights (See Section 3.2.4).

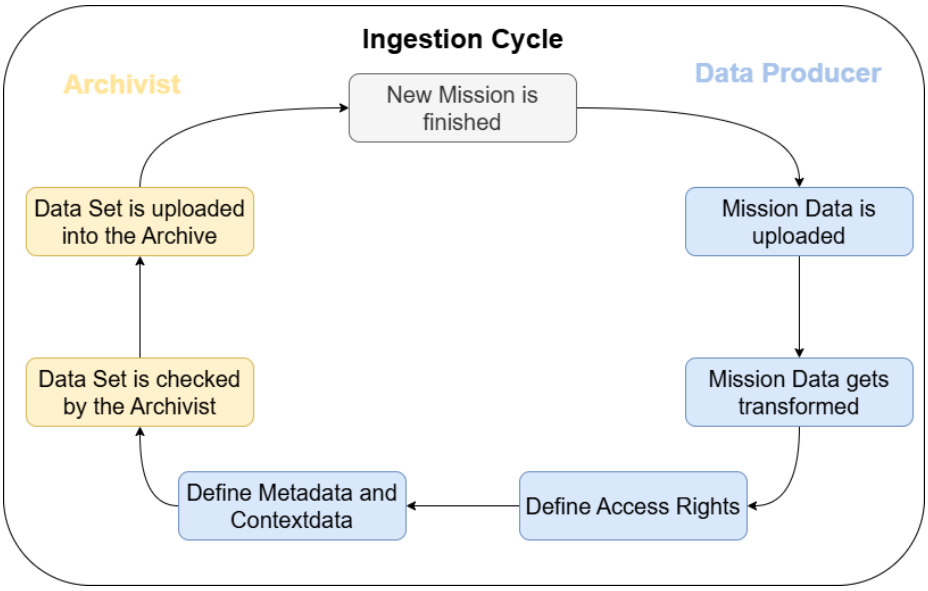


Figure 6.3.: Ingestion Process with Tasks done by Data Producer and Archivist

Also, the duplication mechanisms can be applied similarly. These procedures should be performed by the archivist, who has the required knowledge of the structure and requirements. The following presented example of a flight data archive consists, in addition to a database, also of a file server. This fileserver has to be maintained as well. This includes tasks like checking if the data types stored are still accessible and maintained, and if not, converting these data types to the current standards to ensure the accessibility of this data.

6.2.6. Data Access and Interoperability

As described in Section 3.2.4, an archive has to provide certain interfaces to access and find the data. Especially the discoverability of the data is important. In the context of rocket launches, the data is often highly complex, and it is not trivial to find the relevant data. This complexity arises from the volume and variety of data collected. Furthermore, the data set can be hard to understand. As a result, users can find it difficult to determine what data exists, where it is stored, and how it can be accessed or interpreted.

For example, a user may be interested in analyzing temperature data and is aware of a specific sensor that records temperature measurements. However, several other sensors might also measure temperature at various locations within the rocket, such as inside the payload part or near the engine compartment. The surroundings might heavily influence the measurements. Therefore, metadata, which documents the data, is crucial. For example, would a categorization or key word documentation help to make the data more accessible. Based on this documentation, an interface that allows a search based on this data could be implemented. This way, the user can find all the parameters that measure temperature for a certain mission and then access this data. This helps avoid overlooking additional data sources, which would potentially lead to missing valuable insights.

The metadata does, however, not only help to discover the data but also to understand the context and significance of the data. Sensor outputs might require calibration details and relationships to other data streams. To elaborate further on the temperature example, when accessing the sensor data, it is important to have an easy way to access, for example, relevant structure plans of the rocket to identify the surrounding sensors and components that influence the measurement values. This is critical for accurately interpreting the readings. So linking the related data to, for example, the interface that allows access to the unstructured part of the data set and making these links easily accessible helps make the archive easier to use. This introduces another task for the archivist, which is to periodically check the metadata, like links, for validity and correctness, as stated in the GEO data sharing and management principles [GEO24].

In addition to the various access and discover interfaces, it should also be possible to export the data. This should be possible in different data formats to import the data into different analysis tools to gain deeper insights into the data. Which data formats have to be supported, should be developed in close cooperation with the user community.

To summarize, the archive must provide interfaces to find data and interfaces to access and export the data while providing easy access to related metadata. So to discover, navigate, and understand the available data. A robust metadata management is crucial for these interfaces to work properly. Such features ensure that even complex and interrelated data sets can be accessed and utilized efficiently, enabling users to derive meaningful insights from the archive.

6.3. Prototype for an LTDP Flight Data Archive

Next to all the theoretical considerations and principles in the previous section, the actual realization and implementation of these concepts are as important. From the database to the various import and export functionalities, metadata management, and security implementations.

To show how these different concepts can be implemented and as a proof of concept, a prototype is developed, which can act as a starting point to develop a flight data archive for MORABA and also other organizations with similar activities.

This prototype consists of multiple components. Starting with the database, where the actual data and metadata are stored. This database is the heart of the prototype, which is built to be robust and scalable to handle large volumes of flight and sensor data while remaining as simple as possible. Next to the database, another storage type is added to be as flexible as possible and to ensure that all different kinds of data types can be handled. From structured to semi-structured to unstructured data. This storage type is a file server, where all kinds of data is stored, which does not fit into the database, like images and videos.

The next component of the prototype is the backend server. This server is responsible for handling the connection to the database and the file server and provides an API for services connecting to the archive. Additionally, the server takes care of user authorization, security, and access control. The last component is the frontend, which allows users to interact with

the data archive. This fronted provides functionalities to query the data and export the queried data. Furthermore, it provides an interface where data can be ingested into the archive, including a particular mechanism for specifying the necessary metadata.

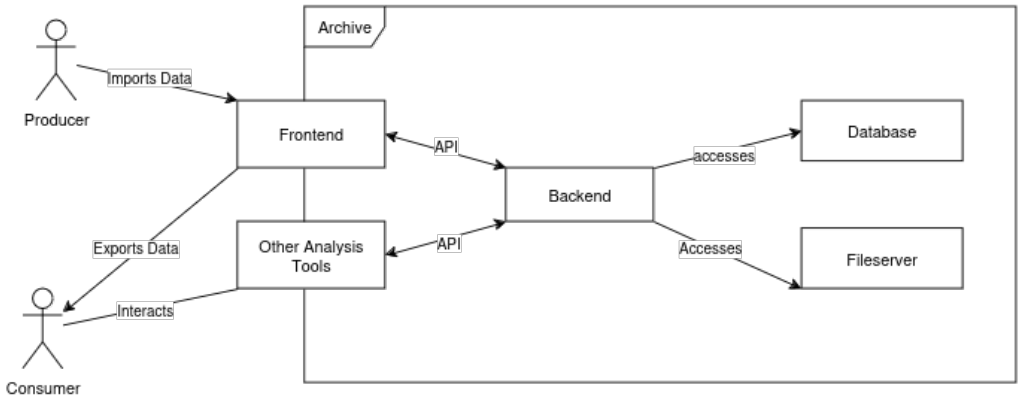


Figure 6.4.: Basic Architecture of the Prototype

In the diagram in figure 6.4, the basic architecture of the prototype can be seen. The two entities interacting with the system are the data producer, which imports new data into the archive using a provided frontend, and the data consumer, who either exports data using the same frontend or can potentially use other analysis tools. The frontend and the analysis tools can connect to the backend server via an API. Then, the backend server communicates with the database and the file server to fetch the needed data.

6.3.1. Storage Entities

As stated in Section 6.1 about the characteristics of a flight data archive, various kinds of data must be stored. This includes structured, semi-structured, and unstructured data. In the following, two different storage entities are presented which can store the complete data set.

Timedata

Starting with the structured data. Most structured data is composed of time series data, which consists of a timestamp and an associated value. Each value is linked to a specific parameter, message, and mission. The combination of timestamp, parameter, message, and mission uniquely identifies each data point. This kind of data includes all sensor measurements, from location to pressure measurements to how many satellites at a certain timestamp are in range. This data is referred to as *timedata* in the following. To store structured data, the most effective way is a relational or SQL database. (See Section 2.3.1 about relational databases).

The database schema for the timedata can be seen in figure 6.5. The schema consists of four different tables. The first one is the *timedata* table, where the value is stored together with the according timestamp and IDs for the related parameter, message, and mission. The remaining three tables are reference tables for missions, messages, and parameters.

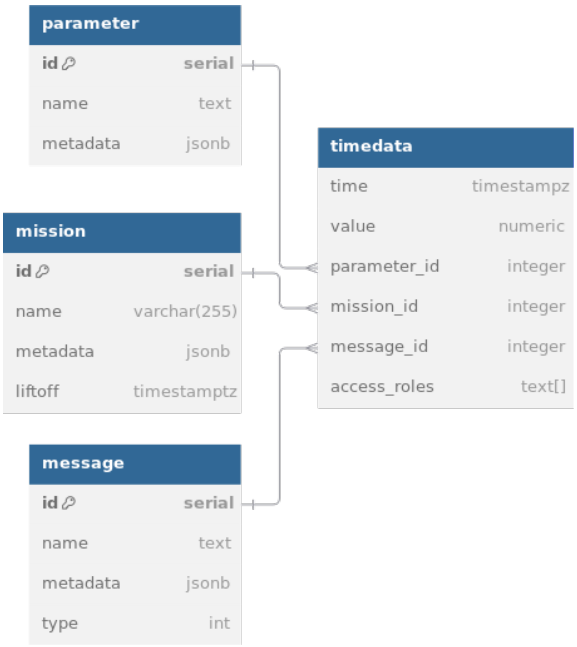


Figure 6.5.: Database Schema of the Timedata

Additional information like the name of the mission table and liftoff time are stored there. All three tables also contain a metadata column. However, this will be described in Section 6.3.1 about metadata.

For implementing the database in the prototype, PostgreSQL was chosen [Gro23]. Firstly, because it is an open-source database management system, which avoids a vendor lock-in. It guarantees ACID (Atomicity, Consistency, Isolation, and Durability), which ensures reliability and data integrity. Then, it provides a strong performance, strong security features, and robust community support. However, the most important feature for the flight data archive is the TimescaleDB extension [Tim23]. This allows having the performance and storage optimization for timestamp data, introduced by time series databases (see Section 2.3.6), while being able to use standard PostgreSQL for the remaining data. This, however, adds additional effort to the import of new data.

Currently, the data comes in the form of SQLite files. These consist of multiple tables. One of these tables represents a device, or sub-device, which sends a collection of related parameters. This is referred to as a message. A message consists of a column with a timestamp and multiple different columns, called parameters, representing measurement values. The TimescaleDB extension utilizes a feature called hypertables to efficiently manage and store time-series data. For hypertables to function correctly, they require a designated timestamp column, which serves as the foundation for organizing and querying time-based data.

In this schema, each column in the hypertable represents a single measurement at a specific timestamp. This decouples the data structure from the structure of the SQLite files, where each table has different amounts of parameters. The inconsistent table structures complicate

data ingestion and querying, whereas the one-measurement-per-row approach standardizes the structure. So, the introduction of new parameters does not require structural changes. Additionally, since each measurement is stored independently, data sets with sparse parameter usage (so parameters only active during specific mission phases) are naturally supported without creating unnecessary empty columns or wasting storage space. Furthermore, this schema allows the use of the extension to its full potential, with performance optimization such as chunking and parallel queries. Analysts can easily retrieve, filter, and aggregate data for specific parameters, devices, or missions across any time range. This flexibility is crucial for troubleshooting, trend analysis, and mission planning, where precise insights are derived from individual measurements.

To maintain data integrity and provide context, in addition to timestamp and value, foreign key relationships are used to uniquely identify each measurement. These foreign keys typically link the measurement to related entities, in this case, parameters, messages, and the associated mission. So, to ingest the data, it has to be transformed from the SQLite schema to the TimescaleDB schema.

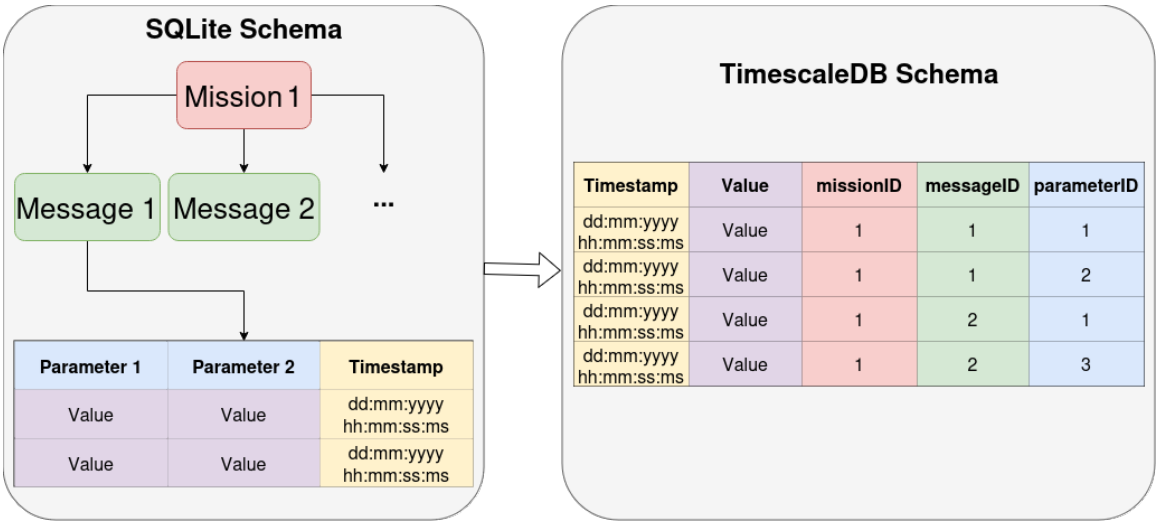


Figure 6.6.: SQLite to TimescaleDB Schema Transformation

Figure 6.6 visualizes the transformation from the schema of an SQLite file to the schema of the TimescaleDB hypertable. On the left side, the hierarchical schema of a SQLite file can be seen where a file consists of one mission containing multiple messages. Each message contains multiple parameters with one timestamp column. The schema of the TimescaleDB table containing the same data can be seen on the right side. There, everything is put into one table, and the timestamp value tuple gets assigned to the according IDs for mission, message, and parameter.

In Figure 6.5, the timedata table is a hypertable from the TimescaleDB extension, while the other three tables are standard PostgreSQL tables. The hypertable has a chunk size of 24 hours, which means there is usually one mission per chunk. This allows fast querying

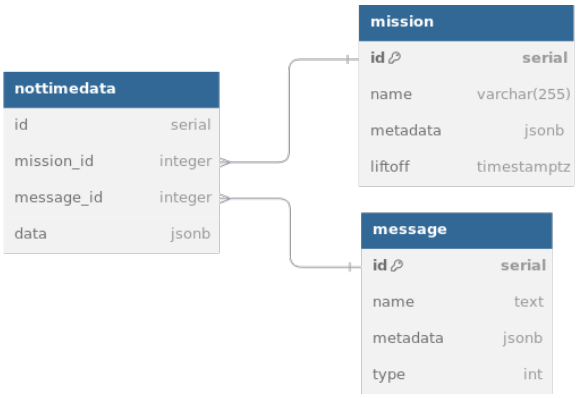


Figure 6.7.: Database Schema of the Nottimedata

over time. The edge cases that more than one mission is on the same day or a mission spans across a day change are rare and, therefore, negligible.

Nottimedata

Next to the timedata, there are also two types of semi-structured data. The first type includes various kinds of lookup and assessment data defined during the data collection. This consists of various metadata as well as message and mission information. This data is stored in an additional table called nottimedata, which is also the preferred name of this kind of data in the following. Similar to the time-series data, this data is also associated with a message and a mission, however, not a parameter. The schema for this kind of data can be seen in Figure 6.7. This data is highly variable and does not follow a certain structure. To handle the high variability and lack of fixed structure characteristics of this type of data, it gets stored as a BJSON column type. This is a PostgreSQL column type that allows the storage of JSON-like data in a binary-optimized format. [Gro23]

This approach provides several advantages. The first one is that BJSON allows storing high-variable data without requiring rigid schema definitions. On the other hand, because it is still stored in a relational table, it benefits from the advantages of relational databases. In this case, PostgreSQL provides extensive functions and operations for working with BJSON columns, enabling complex SQL queries, such as filtering data based on specific keys or values directly within the database. By integrating BJSON into a relational table, the archive benefits from the reliability and robustness of relational databases while accommodating the flexibility needed for semi-structured data and the advantage of not being forced to manage multiple databases. [Gro23]

Metadata

The second type of semi-structured data that must be stored is all kinds of metadata. This includes basic information like the liftoff of a flight, but especially all information needed to ensure long-term data preservation of the stored data. Furthermore, it includes all metadata and context data needed to ensure good discoverability. As outlined in the previous section,

storing the metadata as close as possible to the related data makes sense. Therefore, it is stored in the same PostgreSQL table as the actual data is stored. This can be seen in Figure 6.5 and Figure 6.7.

In these tables, each mission, message, and parameter has a dedicated 'metadata' column, which uses the BJSON data type, as previously described. This approach provides high flexibility for storing diverse types of metadata and is readily adaptable to specific or evolving requirements. Additionally, since it is embedded within the same PostgreSQL schema, it is possible to query metadata and data in a single statement. This simplifies data access and improves the efficiency of data management workflows.

Additionally, each data point's various identifiers and foreign keys can be seen as metadata. The schema preserves the relationships between data points by associating each measurement with foreign keys and linking them to its parameters, messages, and missions. This ensures that all measurements are contextually meaningful and traceable, which is vital for flight data analysis, as measurements are rarely standalone and depend on their operational context.

Fileserver

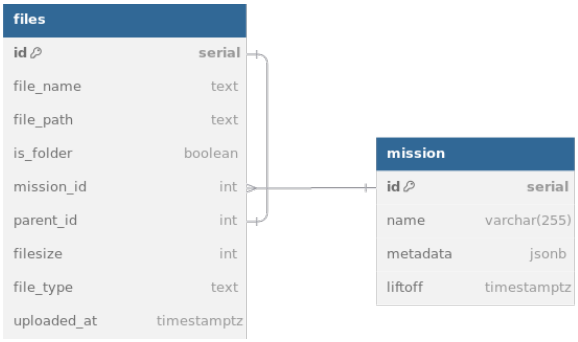


Figure 6.8.: Database Schema of the Fileserver Metadata in the PostgreSQL Database

The last storage entity is where all the unstructured data is stored. This is all kind of data produced during a mission's preparation, realization, and post-processing and is not in a table-like structure. This can be images, videos, and documents. The easiest and most efficient way of storing this data is on a file server. This provides the flexibility needed to efficiently store as many different data formats as possible. In the PostgreSQL database is a table that stores all the metadata for the file server. This includes type, size, upload data, and the path on the file server. Each file is part of a mission inside a parent folder. The schema of the file server can be seen in Figure 6.8

The whole database schema can be seen in Figure 6.9. There, the six tables needed to store the structured and semi-structured data and the metadata for the unstructured data on the file server can be seen.

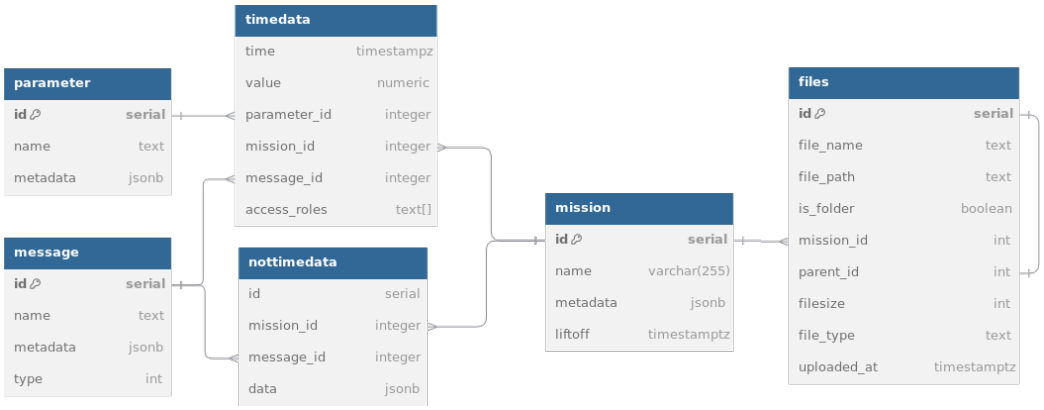


Figure 6.9.: Database Schema of the PostgreSQL Database

6.3.2. Backend

The backend of the prototype connects the developed web frontend and potentially other user tools to the presented storage entities. Thereby, it performs various tasks. The setup is an Express.js setup with an Apollo server [Apo24] for GraphQL [Gra24]. This structure supports GraphQL API for the data handling and uses a Rest API to handle files. GraphQL is used because it offers better response times and greater flexibility compared to a REST API [QnMFGRC23]. REST endpoints handle file uploads and downloads because file transfer is handled better with a REST API. So the backend connects to the PostgreSQL database, and offers various GraphQL endpoints to request and retrieve data from the database.

Authentication and Authorization

As described in Section 6.2, authentication and authorization must be fulfilled to ensure the security and accessibility of the flight data archive. A simple login system is implemented on the frontend to manage user access. This system will securely verify users' identities using the Lightweight Directory access protocol (LDAP) [KV04] authentication. Currently, a local comparison to a hard-coded user database is used for testing. However, in the future, the existing LDAP system of the DLR should be used. Nevertheless, this can be adapted to any existing or newly created LDAP system. In this way, the system can reliably confirm user credentials against an existing directory of authorized personnel, and there is no need to provide its own user management system.

Inside the DLR LDAP directory, all users have assigned roles, which map to their access rights, so which data they are allowed to work with. Once a user has successfully logged in through the LDAP system, these roles are shared with the server. To streamline subsequent logins and minimize repeated LDAP queries, the roles are saved inside the session attributes of the PostgreSQL database upon authorization.

To ensure a user can only see and use data according to his access rights, Row Level Security (RLS) [Pos24] is used. RLS is part of PostgreSQL, which can be described as partial access to a table. The timedata table in Figure 6.5 has a column called access_roles. In this column, the roles that allow access to the data are defined. This allows to specify for

every data point, which roles have access to that data point. This provides a flexible way to hide data from unauthorized access by different attributes like parameters, timestamps, or the value itself. RLS works by defining policies, which define the conditions under which a user can access certain data. In this case, the policy defines that the user's roles must be included in the access_roles field. If a user's roles do not match those defined for a data point, the query will not return that data point, so access to restricted data is blocked. This ensures that users can only retrieve data for which they have explicit permissions, and any attempt to query data beyond their authorized access will return an empty result. [Sha20]

This approach enables a robust, role-based access system that protects sensitive data at the database level, aligning data access with each user's assigned roles.

6.3.3. Frontend

The frontend provides the functionality to upload, download, discover, and get a quick view of the data. It is implemented in React [Rea24] and connects to the API provided by the backend. When opening the web page, the user must first log in using his credentials, as explained in the previous part. After that, he can choose between four pages. The home page, which is the welcome page when entering the web page; the data page, where data can be queried and downloaded; the files page, which is the file explorer part of the web page; the import page to import data into the archive and a page to search for data.

Data Page

The data page provides functionality to query and download different kinds of data. In Figure 6.10, the page can be seen. Starting with the possibility to query the in Section 6.3.1 described timedata. As presented, a data point can be uniquely defined by the mission, message, parameter, and timestamp. Therefore, it is possible to define these four attributes. A search mask consists of three dropdown menus where the mission, message, and parameter can be selected. Multiple selection is possible. For the timestamp selection, there are two different possibilities. Either, two different timestamps can be set, and all data in between is queried - possibly including multiple missions. The second option allows users to specify a duration relative to the rocket's liftoff, defined in milliseconds before and after launch. When multiple missions are selected, this duration applies to each mission individually. For example, if the user selects the first minute of flight, the system will query and retrieve the initial minute of data for each chosen mission.

So once the user selects the option he needs, the button on the lower right can be clicked, an API call with the selected options gets sent to the backend, and the data gets queried. When the data gets queried, in Figure 6.10 in the lower half of the screen, the data is shown in a table-like format. Each queried parameter is displayed with a timestamp, and the associated mission name. This can give quick insights into the data. Additionally, as mentioned previously, it is crucial to have metadata close to the data to adequately describe it and understand the information. Therefore, together with the data, also the related metadata is queried. This metadata, categorized into mission, message, and parameter, can be accessed via a second tab.

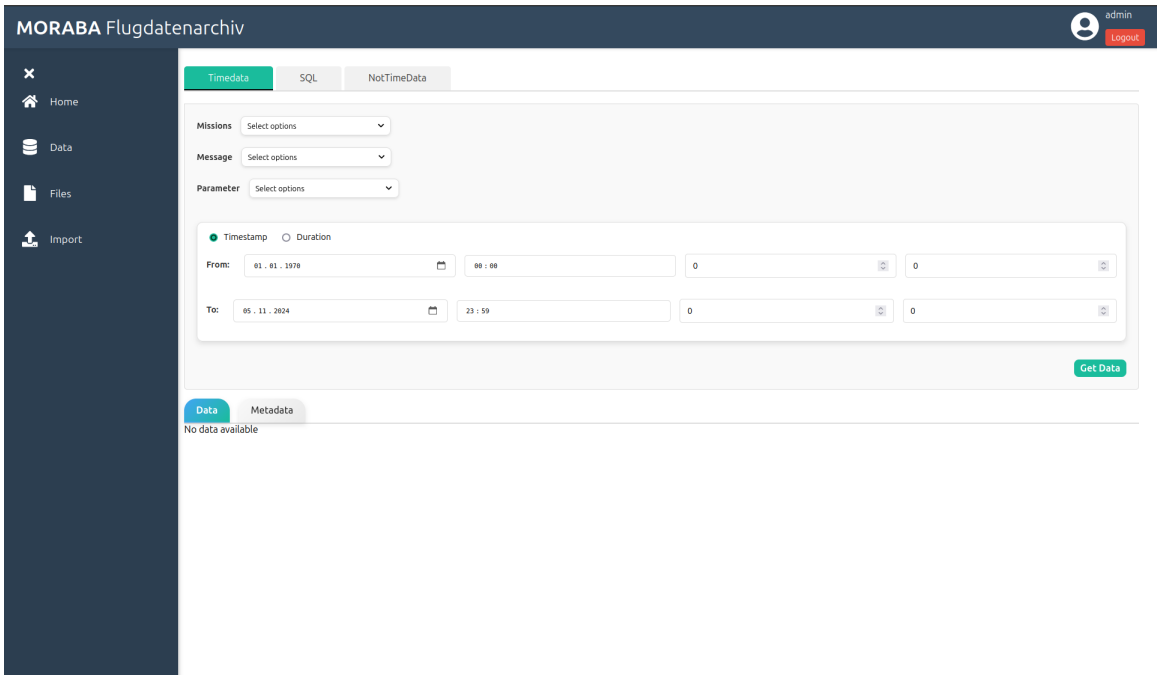


Figure 6.10.: The Data Page of the Frontend

Next to the query possibility for timedata, also the nottimedata can be queried through a similar search mask as described for the timedata. Because of the differences in the structure of the data, the search mask is adapted to the structure, so, for example, there is no possibility to choose time intervals because this data does not have a timestamp. However, the retrieval of the data and the according metadata works in the same way.

Furthermore, when more complex queries are needed, it is possible to write raw SQL statements. This allows users to use the full advantage of the SQL approach with the drawback that knowledge of the schema is a prerequisite. This way, querying data based on the in JSON stored metadata is also possible. These statements run through the same authorization and authentication steps as the other queries.

Files Page

The second page is a page to browse the data not covered by the data page. This is the unstructured data that is stored on a file server. A screenshot of the page can be seen in Figure 6.11.

The page is built like a fil explorer. The files are organized in directories. Each mission consists of one directory and several sub-directories depending on the files stored. Through this page, files can be uploaded, which are automatically added to file storage and to the PostgreSQL database, where the file metadata is saved. The stored files can be downloaded through the UI. New folders can be created when needed, and paths and other metadata are also stored in the PostgreSQL table. Furthermore, files can be uploaded through the according button.

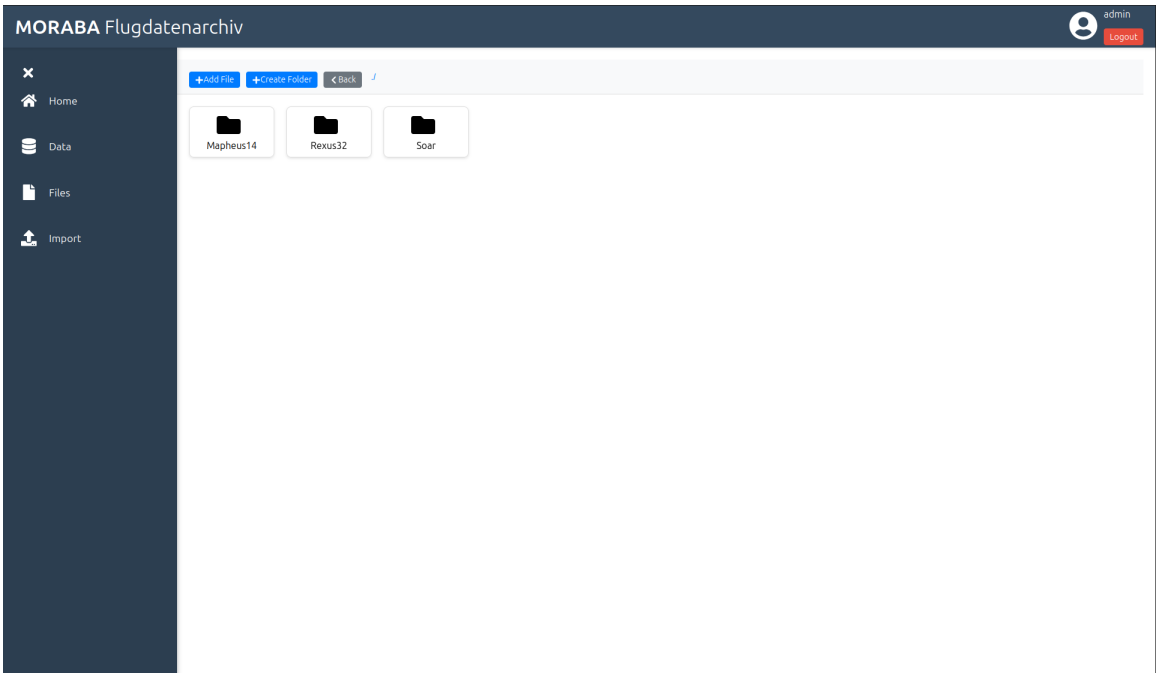


Figure 6.11.: The Files Page of the Frontend

Import Page

The import page defines the upload of new data into the archive. Currently, the end of the data processing pipeline at MORABA are SQLite files. Therefore, the Import page currently supports the upload of SQLite files. However, this can be easily modified. The upload process allows the user to choose a SQLite file he wants to upload. Thereby, a predefined structure of the SQLite file is expected, where one SQLite file represents one mission. It has tables that represent messages, and each column of a table is a parameter. This is the transformation process described in figure 6.6.

Once it is uploaded, the file's different parameters and messages are extracted. Then, all parameters and messages are compared to the existing ones in the archive. If it already exists, its metadata gets extracted and displayed. So, for each message and parameter, the standardized metadata set is displayed, either already filled when the parameter/message exists or the user has to enter the related information.

In addition to the metadata for messages and parameters, metadata for the uploaded mission, like launch location, can be entered. Once all the required metadata is filled in, the data can be uploaded.

In Figure 6.12, a screenshot of this page can be seen. At the top, the SQLite file can be selected, and then, underneath, the level for which metadata should be filled in can be selected. This corresponds to the same structure presented in Figure 6.2. In the screenshot, the Mission tab is currently selected, and therefore, the different metadata for the mission can be filled in.

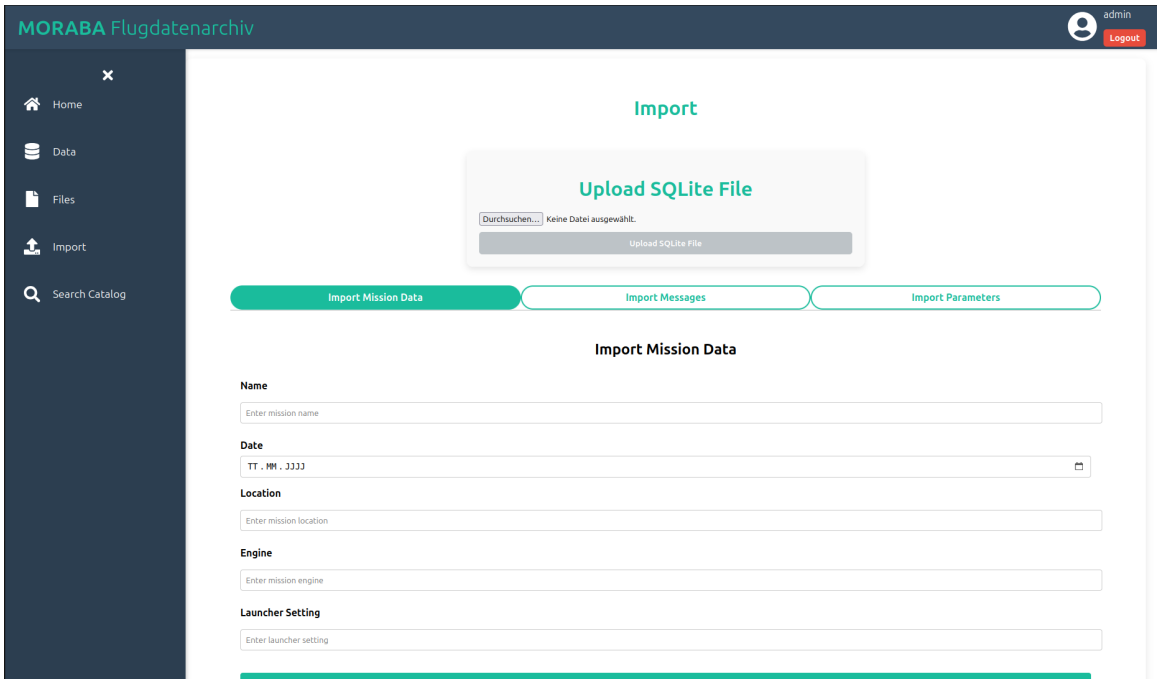


Figure 6.12.: The Import Page of the Frontend

Search Catalogue Page

The search catalog page is the fifth and last page available on the frontend. It provides an interface to find and discover data in the database. Next to finding specific information about one mission or one message, it also provides a mechanism to find correlation between missions. As outlined in Section 6.2.6, it is important to provide a way for the user to discover the data he is looking for, especially in a field with as complex data as rocket launches. While the user might know what he is looking for, he might not know which parameter represents that data set or what other data sources he could consider.

The search page is based on the metadata collected during the ingestion process described in the previous section. This metadata helps filter the relevant data the user is looking for. Thereby, the page provides two different possibilities. The first is to search after missions, and the second is for parameters.

The mission search feature enables users to identify correlations between different missions efficiently. For instance, users can search for all missions that utilize a specific engine at a particular launch site. Without this interface, retrieving such information would require manually reviewing each mission to verify its launch location and engine type. The user can combine different attributes stored in the metadata field to find overlapping mission characteristics. In this way, the user finds relevant missions faster and more efficiently. Then, he can take a closer look at the returned missions to, for example, compare specific parameters across multiple missions that meet the desired criteria.

In Figure 6.13, the mission search part of the page can be seen. Each box at the top of the page resembles one attribute of a mission. So, in each box, one or multiple selections can be made. Based on these selections, the missions get filtered, and those fulfilling these characteristics are returned underneath.

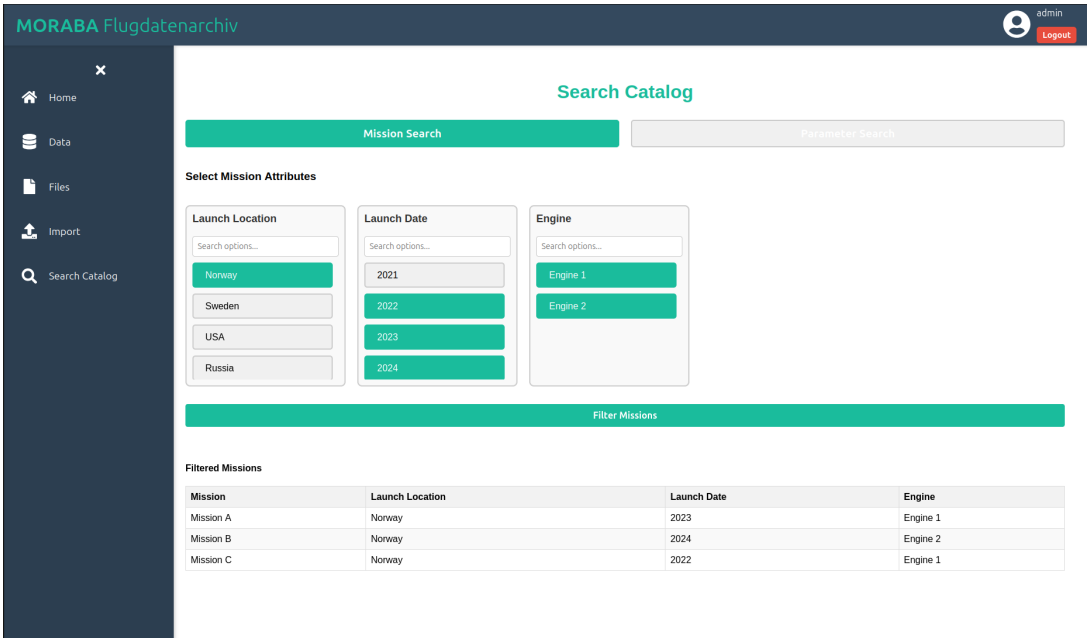


Figure 6.13.: The Mission Feature of the Search Catalogue Page of the Frontend

The second part of the catalog page, the parameter search feature, works similarly but is intended for a different application. The same boxes exist on this part, however, filled with attributes of the parameters, stored in the metadata field of the parameters. The idea of this page is to find data on a specific mission. This directly correlates with the example described in Section 6.2.6. There, a use case is described, where a user is looking for temperature data of a specific mission. However, he does not know which parameters represent the temperature data of the mission and what temperature data exists. So, with the parameter feature part, the user can select the mission and different parameter attributes he is looking for and gets all parameters that fulfill these characteristics returned. In this way, he gets an overview of which data exists. Without the described interface, this information could only be gathered by manually looking through all parameters to determine which store temperature data.

For this page to function as intended, metadata collection must be performed efficiently and comprehensively for every mission stored in the archive. Without the required metadata, the filtering mechanisms will fail, preventing the retrieval of complete mission data or parameter sets and will lead to the potential loss of valuable insight. This also follows Section 2.4, where the outcome of a research on user interactions was that detailed metadata description is vital for users to find the required data.

6.3.4. Use Case Example

This subsection describes a use case of the whole prototype to demonstrate how the different components can work together to form a powerful data discovery and accessibility tool. A constructed example, inspired by a scenario that arose a few years ago at MORABA, was that it should be calculated how much mass for how many kilometers was transported by rockets launched in Sweden in the last five years. At that time, to get this information, all missions launched in Sweden had to be identified by hand, and then the data of these missions had to be found. After that, the parameter that measured the altitude had to be identified, and then the information about the payload mass had to be found, which resulted in reading various postflight reports to find information about the mass. Furthermore, if this calculation should include missions multiple years or even decades ago, this would most likely not be possible anymore because, as outlined in Section 4.2, there is no central storage point for all the mission data.

With the prototype, this workflow is possible in a more efficient and less time-consuming way. The user can use the mission search feature of the search catalog page to identify all missions launched in Sweden in the last five years. Then, he can use the parameter search feature to identify the parameters that measure the altitude of each mission. Using the data page described in Section 6.3.3, he can query the parameters identified in the steps prior. Subsequently, with the same query, the metadata of the selected missions gets queried together with the sensor data. In the metadata tab of the data page, the different masses transported by the rockets can be accessed.

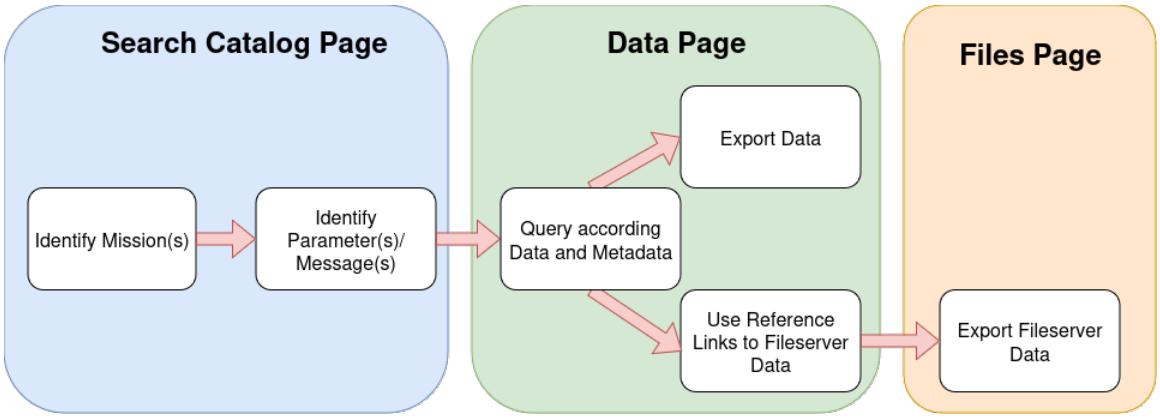


Figure 6.14.: Workflow to Find and Export Data

In this way, with only three steps, identifying the missions, identifying the parameters, and then querying the data and metadata, a user without prior knowledge about the concerning missions can get the information, which was previously only possible through much manual work. Other use cases with a similar approach could include retrieving all temperature data for a specific engine type to conduct an error analysis after a failure. By comparing data from past missions using the same engine type, potential insights into the cause of the issue can be identified. Using the same approach described above, this can be achieved quickly and efficiently without using various tools to gain all the needed information.

Figure 6.14 provides an overview of this workflow, including the option to export the queried data and find and export associated files from the files page. This shows that the prototype follows the recommendation in Section 3.2.4 from the European LTDP Common Guidelines, where it is stated that there should be different access points for the different kinds of data stored in the archive. In this case, this is the search page for the semi-structured data, the file server for the unstructured data, and the data page for the structured data. So three different access points for the three different kinds of data.

The use case example illustrates how the three main pages of the frontend work together to enable an efficient way of finding and retrieving data and to realize the in Section 6.2 present concepts for accessibility and discoverability.

So next to the preservation aspects fulfilled by the prototype, which is the primary goal of the flight archive, it also provides efficient access and discovery properties to work more efficiently with the data. Benefiting and utilizing the additionally collected metadata to enhance data analysis.

Part III.

Evaluation and Discussion

7. Evaluation

Next to the theoretical considerations and practical implementation, the performance of the archive is an important part. Of course, the performance of an archive is less important than that of an operational database, where a lot more queries are done during active working, and waiting times are much more critical. However, the query times of an archive should also be manageable. Various performance tests and evaluations of these tests were performed to analyze this. The test and the evaluation are presented below. The tests were done directly on the database running in a docker container.

7.1. Performance Tests

An important factor for an archive is how the performance changes when a lot of data is in a database. Considering that the archive should last for decades, the performance undergrowth is critical. The difference in system behavior between only a few missions stored in the archive and multiple missions stored in the archive helps to estimate how the system will perform in the future. Therefore, various performance tests of the same queries with different data volumes were performed
Next to query performance, ingestion, and update performance are often critical aspects of databases. However, in this use case, update performance does not matter because the idea is that in the archive, the data that gets injected already was tested for correctness and importance before getting ingested into the achieve, and therefore, only data that belongs in the achieve, is in the achieve.

Performance Environment

To have a solid scientific foundation for the performance test, the following best practices for database performance testing were followed:

A controlled environment is necessary to ensure that the results of the queries are reproducible and reliable. Therefore, the same software and hardware were used, and the potential variations in the results are due to the difference in data size and not the environment. [DHK+85] In this case, the same docker container, with the same configurations and running on the same hardware was used, and therefore there should be no influence of the environment.

However, in a controlled environment, there is also the possibility of influence from other factors. Therefore, performing multiple trials of the same queries with the same data volume makes sense. After that, statistical methods like average median and standard deviation help obtain a more accurate comparison between both. Therefore, every query on every database version is performed 10 times. Additionally, a new session is started before every query to

remove the influence of caching. Furthermore, if the database is freshly restarted, random basic queries get executed to remove any cold start effects of the database. [SPK+22]

Next to the environment and the execution of the tests, the tests themselves are also important. The selection of adequate SQL queries, which represents the typical work done using the archive, helps to get a good overview of the performance when the archive is realized. It is not mandatory to have a lot of different queries, but rather, it is necessary to have the right queries that cover the complete scope of the database. [BD84]

The developed queries for this performance test can be seen in the next Section. Additionally, these queries follow the same schema used by the backend of the prototype to construct queries. So, if the same mission, message, and parameter are selected in the frontend, the resulting query would align with the structure of the queries described in the following. This ensures that the performance evaluations are as representative as possible.

By following these best practices, consistency across the different tests is ensured, and a more comprehensive and accurate performance comparison of the database architecture with different data volumes can be achieved.

Use Cases

Before constructing the queries, it is important to identify the performance-critical part of the architecture. For the presented prototype in Section 6.3, this is the timedata (Section 6.3.1) table storing the time-series data. This is the performance critical table because it will hold millions of rows of data. Additionally, this is the hypertable created with the TimescaleDB extension focusing on the performance of time series data.

The First Query covers the most common use case. This is getting insight into specific data by quering one parameter from one message from one mission. This is the most basic query. The query is tested with and without join operations to test the impact of joining the time series table with the various lookup tables. These join operations are needed to correctly display the names of the parameters, messages, and missions on the frontend. The query without joins can be seen in listing 7.1 and the one with joins in listing 7.2. The parameter was selected arbitrarily, however, it exists for that message and mission.

Listing 7.1: Query 1

```
select
    timedata.time, timedata.value
from timedata
where
    timedata.mission_id =3
    AND timedata.message_id = 27
    AND timedata.parameter_id =71
```

Listing 7.2: Query 1 with Joins

```
select
   timedata.time,
   timedata.value,
   parameter.name AS parameter,
    mission.name AS mission,
      message.name AS message
FROM timedata
      JOIN mission ON timedata.mission_id = mission.id
      JOIN message ON timedata.message_id = message.id
      JOIN parameter ON timedata.parameter_id = parameter.id
WHERE
   timedata.mission_id = 3
   AND timedata.message_id = 27
   AND timedata.parameter_id = 71
```

The Second Query is the next complexity step in data retrieval. Two parameters from two different messages from the same mission get selected. This is a typical use case when data from two sensors from a different device or sub-device onboard the rocket measure the same or related physical values and should be compared. The presented prototype can retrieve the data for such a comparison with one query. Again, the two parameters are arbitrary but do exist. The according SQL query can be seen in Listing 7.3.

Listing 7.3: Query 2

```
select
   timedata.time,
   timedata.value,
   parameter.name AS parameter,
    mission.name AS mission,
      message.name AS message
FROM timedata
      JOIN mission ON timedata.mission_id = mission.id
      JOIN message ON timedata.message_id = message.id
      JOIN parameter ON timedata.parameter_id = parameter.id
WHERE
   timedata.mission_id = 3
   AND timedata.message_id = any(array [26, 27])
   AND timedata.parameter_id = any(array [71, 406])
```

The Third Query tests the performance of getting the same parameters from three different missions. This represents a query where one parameter of three different missions should be compared to get information about the difference between the missions or to find a reason for a failure. Additionally, a timestamp filter is added to use the performance of the PostgreSQL extension TimescaleDB. This timestamp queries data from one minute before liftoff to 20 minutes after liftoff of every mission. The query can be seen in Listing 7.4

Listing 7.4: Query 3

```
select
   timedata.time,
   timedata.value,
   parameter.name AS parameter,
    mission.name AS mission,
      message.name AS message
FROM timedata
      JOIN mission ON timedata.mission_id = mission.id
      JOIN message ON timedata.message_id = message.id
      JOIN parameter ON timedata.parameter_id = parameter.id
WHERE
    timedata.time >= (mission.liftoff - (1 * interval '1 minute'))
   AND timedata.time <= (mission.liftoff+ (20 * interval '1 minute'))
   AND timedata.mission_id = any(array[1,2,3])
    AND timedata.message_id = any(array [23, 15])
    AND timedata.parameter_id = 48
```

The Fourth Query In the first three queries, the amount of data retrieved was unaffected by the ingestion of additional missions because they acted only on the initially stored missions. In contrast, the fourth query represents a more dynamic scenario, retrieving data from newly ingested missions, for example, getting one parameter in the first thirty seconds of each flight. This query will not be frequently used in practice. However, it serves as a valuable benchmark for evaluating the system's performance across varying volumes of data. Furthermore, this query leverages timestamps, the key feature of time-series databases. Using the TimescaleDB extension of PostgreSQL ensures the timestamp-based query is highly efficient. The query can be seen in listing 7.5, where one parameter in the first 30 seconds of the flight gets fetched from all missions.

Listing 7.5: Query 4

```
select
   timedata.time,
   timedata.value,
   parameter.name AS parameter,
    mission.name AS mission,
      message.name AS message
FROM timedata
      JOIN mission ON timedata.mission_id = mission.id
      JOIN message ON timedata.message_id = message.id
      JOIN parameter ON timedata.parameter_id = parameter.id
where
   timedata.time >= (mission.liftoff - (0 * interval '1 millisecond'))
   AND timedata.time <= (mission.liftoff+ (30000 * interval '1 millisecond'))
   and message_id = any(array[21,36,19])
   and parameter_id = 39
```

Test Results

In this subsection, the results of the various performance tests are presented. The according queries can be seen in the previous Section 7.1. There will be three tests of the four queries, each with a different amount of data in the database. The first test had only three missions in the archive and 42 million data points. The second test had 12 missions and 140 million data points, and the last with 840 million data points and 60 missions. The number of data points comes from the estimations made in Section 8.4 to simulate the database after one year for the second test and five years for the third test. The results of each single query can be seen in the Appendix (Chapter V).

The First Test is the test with only three missions in the database, resulting in 42 million data points. Each query is performed 10 times, the number of data points, the average, the median and the standard deviation can be seen in table 7.1.

Query	#Data Points	Average	Median	Standad Deviation
Query 1 without joins	10 993	18.729	18.5925	0.505
Query 1 with joins	10 993	23.5824	23.6625	0.538
Query 2	21 986	39.7825	39.862	0.663
Query 3	3435	21.784	21.633	0.570
Query 4	1347	60.2854	60.3705	1.842

Table 7.1.: Query Results of the Database with three Missions and 42 Million Data Points

The Second test is after one year of data ingestion. After the estimations made in Section 8.4, there will be 140 million data points in the table. (see Table 8.1.) To simulate the different missions performed over one year, the same three missions get ingested multiple times until the according number of data points is reached. However, they get ingested in different chunks of the TimescaleDB table with different timestamps. In this way, no artificial data is inside the database, which could influence the result. In total, there are 12 missions in the database. The results can be seen in Table 7.2.

Query	#Data Points	Average	Median	Standard Deviation
Query 1 without joins	10 993	26.4113	26.372	1.078
Query 1 with joins	10 993	31.792	30.865	2.176
Query 2	21 986	47.316	47.533	0.661
Query 3	3435	28.726	28.759	0.829
Query 4	5388	385.2785	385.5905	1.990

Table 7.2.: Query Results of the Database with 12 Missions and 140 Million Data Points

The Third test is after five years of data ingestion. After the estimations made in Section 8.4, there will be 60 missions and 840 million data points in the table. (see Table 8.1.) As for the second test, the same three initial missions are again ingested into the archive until the calculated number of data points is reached. The according results can be seen in Table 7.3.

Query	#Data Points	Average	Median	Standad Deviation
Query 1 without joins	10 993	75.431	75.361	0.704
Query 1 with joins	10 993	82.212	81.997	0.882
Query 2	21 986	108.058	105.36	5.782
Query 3	3435	90.887	91.135	0.976
Query 4	25 746	31 538.710	31 406.3445	358.852

Table 7.3.: Query Results of the Database with 60 Missions and 840 Million Data Points

7.2. Performance Evaluation

In this subsection, the test results of the previous section are analyzed and interpreted with a focus on the database and, in particular, the Timedata table storing the time series data. The focus is on the database because it is the bottleneck of the whole system regarding performance. The backend and frontend are also performance relevant. However, at the moment, these are only prototype tools to show how interaction with the database can look. Therefore, they are not built on performance and should not influence the result. Four different tests with three different table setups were tested. These tests aimed to investigate the table's behavior with different data volumes and simulate the performance after years of data ingestion.

Stability

Before analyzing the differences in query time between the table setups, insights can be derived from the average, median, and standard deviation. As it can be seen in Table 7.1, 7.2 and 7.3, the average and the median are for all queries close together. The average difference between the median and the average is 0.62%. The average of the standard deviation is 1.3711 ms. While no statement about the absolute performance can be deduced from these numbers, it shows that the setup has a stable and consistent performance. A small difference between the average and the median suggests that the data is fairly symmetric without extreme outliers. However, the fourth query in the third test has a noticeably higher standard deviation, making it a clear outlier compared to the other results. While the deviation for this specific query is higher, it remains proportional to its overall query time, aligning with the relative stability observed in the rest of the tests. As mentioned in the query description in Section 7.1, this query follows a different approach than the other three queries. The retrieved results scale with more missions ingested into the archive because data from every mission is queried, which results in a more complex query and a higher query time. With a more complex query, the standard deviation naturally also rises.

To summarise, these findings suggest, that the current configuration is well-balanced and capable of maintaining predictable query performance, also with increasing size of the database.

Absolute Performance

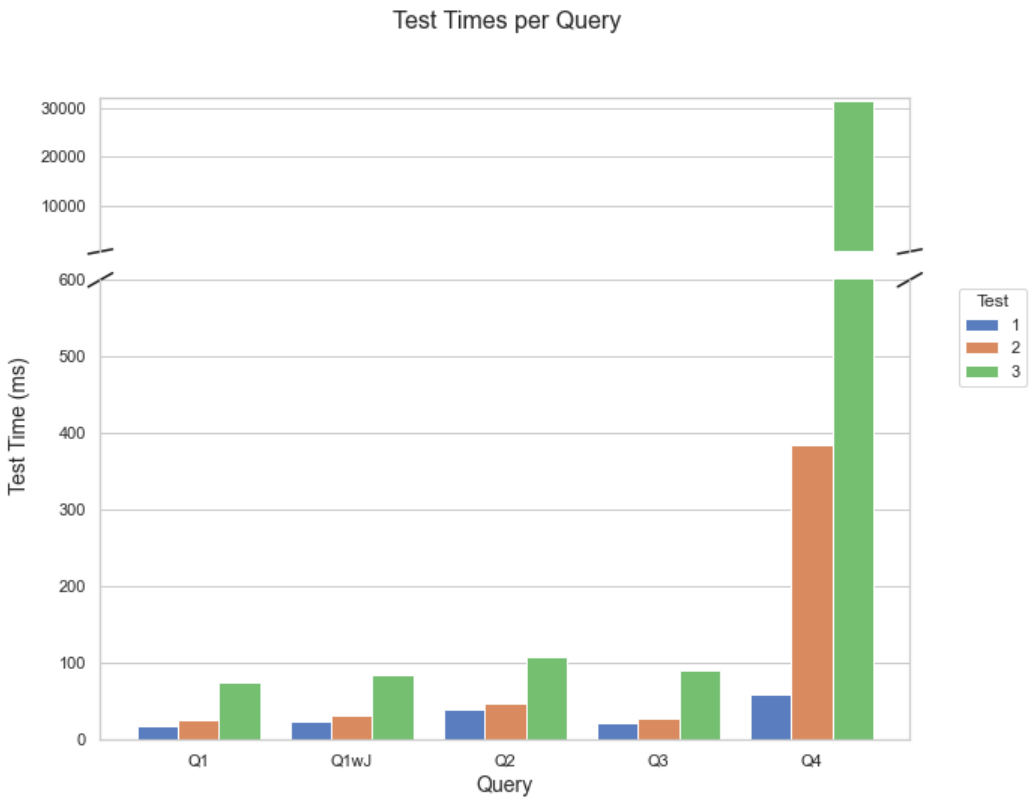


Figure 7.1.: Average Test Time per Query in ms

Figure 7.1 shows the average query run times per test and query. The values are retrieved from table 7.1, 7.2 and 7.3. For each query, three bars resembling the three different tests at the three different states of the database can be seen. The first test is blue, the second orange, and the third green. On the x-axis, the different queries are shown, and on the y-axis, the according run time. There is a gap between 600 and 10000 ms between the five different queries to allow for a meaningful display. For the first three queries, including the first one with and without join operations, the average return time is below or close to 100 ms. This is a response time perfectly acceptable for a data archive. However, it can be seen that the amount of data stored inside the archive impacts the return time Because, for every query, the query time is higher with a larger data volume. How high this impact is will be discussed in the following section. However, as mentioned, even after five years' worth of missions, the return time for these basic but often occurring queries is below or close to 100 ms, which does not influence the usability of the archive.

Only the fourth query takes longer. As described in Section 7.1, this query returns one

parameter of the first thirty seconds of each mission. So, with an increasing number of missions stored in the archive, the number of returned data points also increases. The last test of the third query takes the longest as it retrieves a single parameter from the first thirty seconds of sixty missions. While this is an uncommon query with limited practical relevance, it demonstrates that the system can handle queries for which the archive is not specifically optimized. Therefore, a query time of around thirty seconds is acceptable.

In Figure 7.1, the first two queries resemble the test times of the first query without and with join operations. Thereby, the query with join operations is, on average, 18% slower than the query without. However, the difference decreases with the increase in data volume. In the first test, it is 25.9%, 20.3% for the second test, and 9.0% for the third test with the highest database. This shows that the join operation only has a marginal impact on the query time, which also decreases in relation to increasing query time caused by increasing data volume.

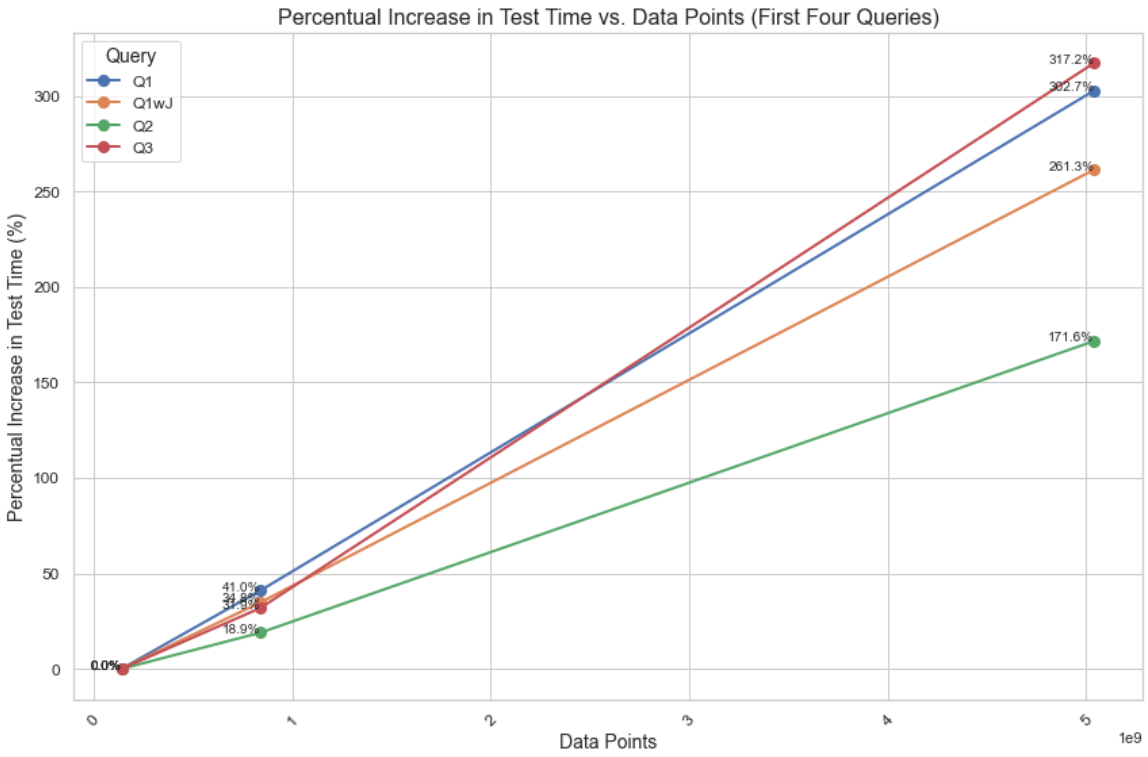


Figure 7.2.: Percentage increase of Query Time

In Figure 7.2, the percentage increase in test time on the y-axis to the according data points on the x-axis can be seen. The second and third tests are compared to the first. So the first percentage values are the increased query time due to the increase from 42 million data points to 140 million data points. The second value is the result of comparing query times with 42 million to 840 million data points.

After one year, the data points increase from 42 million to 140 million. This is an increase of around 333%. Compared to the increase of run time for the four different queries, it is 18.9% for Q2, 33.9% for Q1 with join operations, 34.8% for Q3, and 41.0% for Q1. The discrepancy between the growth in data volume and run time increase suggests that the system scales relatively well with data growth. The run time does not increase proportionally to the data growth. In detail, there are different levels of sensitivity to data growth. The lowest increase is by the second query, which is the slowest of the first four tested queries. The highest increase is for the first query without joins, which is the fastest. This suggests that inherent complexity in execution results in a high baseline run time but relatively less affected by data growth.

The third test was performed after an increase of the data volume by 2000%. This resulted in an increase of run time by 171.6% for Q2, 261.3% for Q1 with join operations, 302.7% for Q1, and 317.2% for Q3. This result is similar to the one between the first and the second test. Also, these query times do not increase proportionally to the increase in data volume. This behavior is critical for ensuring scalability in scenarios where data volumes are expected to grow significantly.

Latency Growth Rate

Another metric that can be discussed is the latency growth rate. This rate tells how much latency increases on average per ingested row, and measures how it reacts to the ingestion of data. By dividing the difference in latency by the difference of data points, the average additional latency per ingested data point can be calculated. In Table 7.4, the latency growth rates for the first four queries between tests one and two and tests two and three can be seen.

Query	**Test 2**	**Test 3**
Query 1	$7.84 \cdot 10^{-8}$	$7.00 \cdot 10^{-8}$
Query 2	$8.38 \cdot 10^{-8}$	$7.20 \cdot 10^{-8}$
Query 3	$7.68 \cdot 10^{-8}$	$8.67 \cdot 10^{-8}$
Query 4	$7.08 \cdot 10^{-8}$	$8.89 \cdot 10^{-8}$

Table 7.4.: Latency Growth Rates for different Queries in Test 2 and Test 3 in ms per Data Point

Interestingly, it can be seen that for query one, undependable if with joins or without, for a higher database volume, the latency growth rate decreases, whereas for the third and fourth queries, it increases with a higher database volume. However, an average latency growth rate of $7.84 \cdot 10^{-8}$ ms per data point, which is around 0.78 ms per one million data points, is an acceptable performance loss.

This result should be interpreted cautiously, as it is highly dependent on the database's environment. The test was conducted on a conventional laptop within a Docker container. Using more advanced hardware would likely enhance performance. Additionally, as discussed in Section 3.2.4, both hardware and software should be regularly maintained and upgraded

to align with the latest technological advancements. This implies that as the archive ages, the continuous improvement in hardware capabilities will likely lead to better performance over time or a smaller latency growth, respectively.

In summary, the evaluation of the database performance using various queries and configurations presents its stability, scalability, and overall suitability for handling the demands of a growing flight data archive. The results show that the system maintains consistent and predictable query behavior with an acceptable increase in run time when significantly increasing the data volume.

More complex queries show a higher run time but still demonstrate the system's capability to handle uncommon but computationally intensive scenarios. An average latency growth rate of 0.78 ms per million data points is a reasonable performance given that the testing environment is running in a docker container on a conventional laptop. In conclusion, the performance tests show that the database concept with PostgreSQL and TimescaleDB seems well-suited for the long-time preservation and accessibility of mission data, assuming the system is running on suitable hardware and in addition, the remaining parts of the archive like backend and frontend do not introduce additional bottlenecks or slowdowns.

8. Discussion

In this chapter, various decisions and aspects of the thesis are discussed, including the content of a flight data archive, the metadata, the choice of storage entities, and the operational improvement a flight data archive can provide.

8.1. Content of a Flight Data Archive

The content of a flight data archive is the heart of the whole system. All requirements are derived from what is stored inside the archive. The content described in the last sections is derived from the purpose and the use of a flight data archive. Thereby, certain decisions were made concerning topics like at what point of the mission life-cycle the data should be ingested and who the primary users of the archive are.

One of the primary objectives at MORABA was to improve mission data management by implementing a centralized data management system. However, the different tasks the data management system should fulfill had to be defined. When discussing the various system requirements with the scientists, it became apparent that a clear distinction must be made between an operational and a preservational system. This defined, at the same time, also, what data should be stored in a flight data archive.

The separation has to be made to ensure the efficient solving of the different purposes by the two systems. An operational database should support activities that require immediate or rapid response, such as flight monitoring, anomaly detection, and timely decision-making. So, it focuses on performance, reliability, and immediate access to current data. On the other hand, a preservation archive has to ensure long-term storage, accessibility, and usability of historical mission data. It focuses on data integrity, format standardization, and long-term maintenance. Separating these archives allows each to be optimized for specific purposes without imposing conflicting requirements. These conflicting requirements include the availability of current data as fast as possible for the operational system while only storing high-quality, validated, and curated data in the long-term archive. This combination is only possible with compromises for one of the two. Separating these databases ensures the operational system is not burdened by the overhead of maintaining a massive historical data set, which has no use for the mission operations. At the same time, it allows extensive data validation and metadata collection to ensure data integrity for the data archive because the availability of live data is not needed. The same approach was taken by the PSA, described in Section 3.1.1.

So, it was decided that no current data would be stored in the flight data archive. The data should be ingested into the data archive when the mission is completed. This ensures that data is as correct and complete as possible, and therefore also, no updates to the data should be required. At the same time, this ensures that the necessary metadata is available

when uploading the mission data to the archive and can be ingested together with the sensor data, so it can be ensured that the related metadata is always available in the archive.

8.1.1. Combination of Data Archive and Operational Database

The previous section emphasizes the importance of maintaining a clear separation between the operational database and the flight data archive to meet their distinct purposes. However, to improve usability and eliminate the need to interact with two separate systems, the operational database could be integrated into the frontend of the flight data archive.

This would have the advantage that the user experience is streamlined because they could access both systems through a unified interface, which simplifies the workflow when working with historical and current data. This would allow users to retrieve operational and archived data with a single query, reducing the time spent navigating different platforms. Furthermore, a unified interface would provide a consistent user experience with the same tools for querying, visualizing, and exporting data.

However, it is important that despite integrating both systems into the same frontend, they should maintain their distinct purposes. This implies a higher complexity of the backend as a consequence. Query routines must be implemented to direct requests to the appropriate database. The operational database would need a design and structure similar to the presented concept of a flight data archive to simplify the interactions. Additionally, this integration adds complexity to maintenance and scalability. Changes to one system must ensure compatibility with the other. Additionally, distinct uploading interfaces must be introduced to prevent confusion between the two systems. Clear separation and safeguards should be established to maintain the integrity and usability of both databases within the integrated framework.

In conclusion, while integrating the operational database into the frontend of the flight data archive would enhance usability, it requires thoughtful design and implementation to preserve the distinct purposes and reliability of the two systems. So, it has to be decided whether the gained usability justifies the additional overhead of implementation and maintenance work.

8.2. Metadata

Metadata, which describes all kinds of data, which is not measurement data or images and videos, is crucial for the whole data archive. As outlined in the guidelines presented in Section 3.2, metadata and context are fundamental for the preservation and discoverability of the system.

The capture and storage of the required metadata depends on the use case however it can get very complex quickly. The idea is to have a complete data set, making it possible to access, understand, and analyze the data. This is a complex task. To construct a complete data set, metadata describing this data must be collected for each parameter, message, and mission. This is a very labor-intensive work. For example, three example missions from MORABA consist of 2514 different parameters. Furthermore, with the increasing sensor amount, this amount will increase drastically. Moreover, for each parameter, multiple metadata information has to be collected. This includes general information about a parameter used multiple times for different missions, but also mission-specific parameter

information, such as calibrations. This makes the collection and storage much more complex. With the unique characteristics of rocket launches and the differences between the rockets, automatic metadata creation is difficult. Therefore, it is often necessary to collect the metadata manually. For metadata of this scale, it is crucial to define a data set that is as concise as possible while still encompassing all essential requirements.

This metadata should include, among others, the data processing pipeline of the data. This describes the steps from the collection until the data is imported into the archive. It is important to trace possible errors during the process. The software and hardware versions involved in the process must be collected for this. Many entities are involved in this process. This includes devices on board the rocket and different data-handling software. The collection of this data is a labor-intensive and complex task.

Furthermore, the data processing pipeline is often not faultless because the processing entities are subject to development and research as well. Then, often, there is also human interaction with the data. Also, these interactions have to be documented to retrace what has happened to data to be able to, for example, search for a potential error. While this is not directly the task of a flight data archive, it is crucial for its functionality.

A follow-up problem that emerges is the storage concept of the metadata. This depends highly on the scale and complexity of the data set on which it is agreed. The presented prototype consists of multiple BJSON columns where this data can be stored in a structured way by reusing the already existing schema while maintaining flexibility for the type and size of the metadata. However, this setup limits the amount of metadata stored per data point. A BJSON field can store a maximum of 255 MB [Gor21]. However, a metadata structure requiring 255 MB of data for a single data point, such as a message or parameter, is highly unlikely to be practical. Such a structure would have a level of complexity that may no longer be sustainable. The metadata columns in the various tables are intended to store a standard set of metadata applicable to all data sets. No complex or nested structure, while possible, should be stored there. Then, a more powerful concept might be needed.

8.3. Choice of Storage Entities

Various database concepts were presented in the technical and theoretical background discussed in Section 2.3. Ultimately, for the prototype, PostgreSQL in combination with a file server was chosen for several reasons.

A data warehouse was not selected because it is optimized for analytics and structured data, making it unsuitable for mixed workloads. Additionally, its high costs and the increased complexity of its infrastructure are not justified without clear benefits for this use case. Similarly, data lakes were decided to be unsuitable due to their reliance on specialized query tools, which add unnecessary complexity. Their unstructured nature also complicates the storage and management of millions of data points, which already have a given structure. Although multi-model databases inherently support various data types, they do not provide additional value compared to the chosen approach in this context. Furthermore, they introduce unnecessary complexity for a use case that does not require graph, document, or other specialized models.

In conclusion, PostgreSQL was chosen because it efficiently and reliably handles structured data while being simple, cost-effective, and offering proven performance, ease of integration, and flexibility for future needs.

8.4. Data Volume

The goal of the archive is to store data over a long period. Therefore, the architecture must be capable of storing a huge amount of data. Two types of data have to be differentiated. The time series data are stored in the TimescaleDB database, and although it is a lot of data, it is the smaller amount in terms of bytes. The second type is all the data stored on the file server. These are images, videos, and other documents that make up the main proportion of the data volume of one mission. In the end, however, as outlined in Chapter 7, the performance-relevant part is the time series data. This data will be queried regularly. Moreover, much more complex queries are carried out on this data than on a simple file system. Therefore, it is important to know how much data there will be in the future.

The sample data to build the prototype consisted of three missions with 42 million data points. Each year, between five and fifteen missions are performed, so on average, around ten missions a year would make 140 million data points in the first year. However, the amount of data per mission will also change. Kim Fowler conducted a survey to predict the future developments in sensor technology [Fow09]. The outcome was that the number of sensors (and therefore also parameters) in five years would grow between 20% to 50%. The survey had participants from industrial fields as well as academics. Considering that the technology life cycle for space missions is slower than in other scientific sectors, the lower end of the prediction is used for this estimation. Furthermore, the sensor resolution might change in the future. However, for this rough estimation, this is not considered.

In Table 8.1, a rough estimation of the expected data points can be seen. The estimated 140 million data points per year can be seen in the first year. Then, the estimation for the five and ten-year mark is calculated with the formula of the current number of data points times five years times 1.20, reassembling the growth of sensor data volume per 5 years.

Time	Number of Data Points
1 year	140 000 000
5 years	840 000 000
10 years	5 040 000 000

Table 8.1.: Estimated Number of Data Points

As seen in the Table 8.1, the amount of data points and, therefore, the volume increases drastically. This is only after 10 years when the archive should last for decades. Therefore, also if performance is not the primary goal of the archive, it is still important to keep the achieve functional.

8.5. Operational Improvement

Next to the preservational advantages, other advantages were also created by introducing the concept of a flight data archive. As described in Section 4.2, no central data storage solution currently exists. Without it, working with data from different missions is difficult. Even if the data of two different missions is available, they do not share a database, and therefore, the information has to be extracted from multiple files. For example, additional metadata like calibration values are often stored in different places. This makes working with the current data and historical data for comparisons a labor-intensive task.

With the introduction of a flight data archive, this changes. Parameters of two or more missions can be extracted with a simple query and without the need to access multiple files. The related metadata needed to understand and use the data are returned together with the related data. This makes the process of working with the data much faster and much more efficient. At the same time, the workflow is simpler and easier to understand without additional knowledge about the structure of the context of the data.

In addition, the inclusion of the file server into the frontend helps to make the unstructured data, like videos or documents describing the structure or architecture of a rocket, much more accessible. It reduces the time needed to search for this data. Furthermore, the possibility of searching data based on the metadata through the Search Catalog page concept makes the data even more accessible and drastically reduces the knowledge needed as the use case scenario in Section 6.3.4 shows. In total, the usability and versatility of the archive improves the overall operational capabilities.

Part IV.

Future Work and Conclusion

9. Future Work

In this chapter, the future work for the presented project is discussed. This includes primarily the realization of the provided theoretical and technical results and recommendations on how to further improve the flight data archive.

9.1. Realisation of the Prototype

Over the last sections, theoretical and practical considerations and recommendations were developed and presented on implementing a long-time data archive for flight data from sounding rocket launches of MORABA. The logical follow-up to this is the realization and implementation of these concepts. Thereby, various considerations have to be made when realizing the project.

9.1.1. Hardware Consideration

The prototype was implemented and tested in a development environment, running in a docker container on a standard laptop. This setup was sufficient for proof-of-concept testing, enabling the evaluation of system functionality, data handling workflows, and performance metrics under controlled conditions. However, this does not reflect the requirements of a fully implemented flight data archive for operational use.

The four different components of the described flight data archive have different requirements. The file server must be capable of securely storing large volumes of mission data over extended periods. To mitigate data loss, the in Section 6.2.5 and Section 3.2 recommended security features like data duplication should be implemented.

Similar requirements are needed for the database hardware, with the additional requirement of optimized and fast hardware to handle complex and concurrent queries. This ensures that the database has a low-latency data retrieval and high throughput.

The backend server requires sufficient processing power to handle API requests, process data transformation, and manage communication between storage entities and frontend applications.

The frontend has to be hosted on a server with adequate bandwidth and reliability to allow remote access by multiple users. Furthermore, the whole system should be well connected to the network infrastructure of the organization or company. The frontend should be accessible internally as well as externally if possible. This ensures that the archive users can work efficiently with the archive independently if he is at home, in their office, on a business trip, or like, in this case, at the launch facility of a mission. Accessing the archive remotely provides significant benefits, particularly during missions.

Access to data from past missions can be crucial for comparing and identifying patterns. For example, if anomalies or errors arise during a mission, historical data can offer insights

into similar issues encountered in previous missions and how they were resolved. This capability supports troubleshooting and enhances decision-making by leveraging the information and knowledge from earlier missions. Such accessibility ensures that the archive is not just a repository for preservation but also a powerful tool for operational support and continuous improvement.

Finally, the hardware should be chosen with scalability in mind, as the data volume will grow significantly, as outlined in Section 8.4. By transitioning from a development setup to specialized hardware, the archive can achieve its key goals.

9.1.2. Meta Data Collection

As described in Section 3.2.4 and 6.2, metadata is vital to ensure long-term preservation and discoverability of the content of the data archive. It was pointed out that defining a standard meta- and context data set is important to ensure comparability and continuity across all missions. This standardized set has to be defined before the first mission gets ingested into the archive.

To find out which data is needed, an approach could be to use the information presented in 4.2 and 6.2, which defines the goals and purpose of the archive. With these requirements, future archive users should be asked, in the case of MORABA the scientist working there, what they think is needed to fulfill the requirements and what metadata they use in their daily workflow. This includes data from other sources and internal knowledge gained through their work and involvement in various missions. In this way, a complete set of metadata, which is needed to understand and work with the data, can be collected. Especially the data and knowledge about the data experienced scientists use are essential. This knowledge might not be available in a decade because either the scientist left the organization or the knowledge will be forgotten over time.

Once this standardized data set is defined, it has to be collected for every mission. As mentioned in Section 8.2, this is a labor-intensive task. So, at least partly automation of this task is preferable. This can be achieved by extracting the necessary information from various documents or configurations. An example would be flight requirements plans created before the rocket launch, which include information about the requirements of a flight and basic information like launch location. Another document that can be used is post-flight reports, created after the mission's launch and analysis. These are documents used and created by MORABA for every launch, other organization might have different procedures, however similar documents will be available.

Other information, like software versions of the data processing pipelines or hardware versions of sensors, can be extracted from configuration files or similar. Nevertheless, in the end, a part of the metadata collection has to be made manually. Some information might not be documented, or the documentation may not follow a standardized procedure and structure. Therefore, it is hard to automate the extraction of the information. Nevertheless, this task is important to fulfill the requirements for the archive and to follow Palaiologk et al.'s recommendations that proper metadata collection costs 30 times less than poorly collected metadata in the long-term [PETS12].

9.1.3. Further Improvements

Next to the hardware considerations and the metadata collection process, the prototype needs some further features and improvement to fulfill the in Section 6.2 presented concepts.

Starting with finishing the implementation of the upload and search catalog process. For both, the definition of the standardized metadata set is needed. It is needed for the Import page because it has to be specified which metadata must be filled in for a newly uploaded mission. It is needed for the search catalog because the attributes are defined based on the metadata, after which the different missions and parameters can be categorized.

In Section 6.2.4, the concept of a staging area is introduced. This staging area has to be set up. This means that after a data producer has uploaded all the data for the mission and collected the necessary metadata, he finishes his upload. However, before the data is uploaded, it should be stored in a staging area where another entity, for example, the archivist, can review the uploaded data to check for correctness and continuity. After this, he confirms the data, which is uploaded into the archive. In addition to the staging area, an interface must be built to define specific access rights during the upload process. This interface needs the functionality to define access roles based on data details, like parameter names and timestamps. The guidelines outlined in Section 3.2 recommend implementing a logging system to enhance the usability of the archive. This system would display updates on the home page, informing users about recent changes and improvements made to the archive.

Furthermore, the export capabilities of the archive should be further enhanced to serve a wider variety of file formats. Currently, the data can be exported in txt or csv format. Different format types like HDF5 should be supported to enhance the usability and accessibility of the archive. The formats that should be supported should be defined together with the archive users to allow the best possible inclusion in the existing software environment.

9.1.4. After the Archive

While the archive itself allows a huge gain of accessibility, usability, and preservability of the flight data, it also allows different tools to connect to the interfaces of the archive or use the data stored in the archive once it is implemented. The frontend retrieves the data using the backends API. However, other tools can also connect to the provided API, enabling use cases beyond the current functionality. For instance, a data visualization tool could connect to the API to create advanced graphical representations of mission data, enhancing analysis and interpretation.

Furthermore, by creating a flight data archive, data from various missions is stored in a standardized way. This enables the possibility of machine learning or AI tools to automate, for example, data analysis or pattern detection on the data stored in the archive, which could lead to additional insights into historical as well as operational data. For example, revealing trends or correlations previously difficult to identify when the mission data was not stored in one place.

In conclusion, the modular and open design of the archive ensures that the archive not only serves as a preservational tool but can also serve as a basis for other tools or functionality to further enhance the data management and processing in the organization.

10. Conclusion

This thesis showed how a flight data archive for sounding rocket missions can be effectively implemented. The focus was on the key aspects of preservation, usability, accessibility, and discoverability. Furthermore, it examined the key metadata needed to ensure the long-term usability of sensor data from rocket missions and how these theoretical considerations can be practically implemented.

To achieve this, an overview of the basics of data storage and database systems was given to offer a technical and theoretical background. This foundation established the prerequisites for understanding the complexities of long-term digital data archiving. Furthermore, the aspects of archiving digital data were introduced and presented in detail. Various guidelines for achieving long-term preservation of scientific data exist for earth observation missions. These well-established concepts were analyzed and adapted to the specific requirements of short-duration space missions, forming the scientific basis for the design of a flight data archive.

These concepts were then realized and implemented in a prototype. This prototype consists of two storage entities: a relational database that handles structured and semi-structured data, such as sensor readings and mission-related metadata. The second entity is a file server for all unstructured data like images or videos. Furthermore, the prototype includes a server backend, which manages the storage entities and provides an API for frontend tools to connect to. The backend incorporates user authorization and access control mechanisms, ensuring that data is both secure and accessible to authorized users.

A frontend application was developed to show how the accessibility and discoverability concepts can be realized. The frontend provides functionality to discover, access, import, and export the data stored in the archive.

This work connects theoretical guidelines and practical implementation. The prototype serves as a proof of concept, demonstrating the theoretical concepts in a practical way. Furthermore, the modular design of the archive allows for scalability and adaptability, ensuring it can evolve alongside advancements in technology and the changing needs of the space research community. It shows the key aspects to consider when developing a flight data archive, as well as why such an archive is important. The archive plays a vital role in long-term data management by ensuring no valuable data is lost and preserving knowledge. In addition, the archive can improve the efficiency of working with data from past missions and make these accessible and usable.

In conclusion, this thesis has demonstrated that the integration of preservation, usability, accessibility, and discoverability principles can lead to the development of a robust flight data archive for sounding rocket missions. By addressing both the theoretical and practical sides of data archiving, a framework for ensuring mission data remains accessible and useful for the future is provided.

Part V.

Appendix

Test results

First Test with three Missions in the Archive

Execution	Query 1 wo Join	Query 1	Query 2	Query 3	Query 4
1	18.538	23.877	39.959	22.381	55.832
2	18.647	23.528	40.866	21.685	62.089
3	18.357	23.762	40.346	22.698	62.189
4	20.037	24.154	39.847	21.444	60.500
5	18.290	23.740	39.835	21.581	60.148
6	18.273	23.510	39.929	20.836	60.241
7	18.487	22.770	38.267	21.486	62.495
8	18.857	24.376	39.052	21.305	59.968
9	19.150	23.585	39.847	21.823	58.832
10	18.654	22.522	39.877	22.601	60.560

Table 10.1.: Results for the four Queries and additionally the first one without joins

Second test on database with 140.134.615 datapoints

Execution	Query 1 wo Join	Query 1	Query 2	Query 3	Query 4
1	25.519	31.023	46.092	28.543	387.347
2	26.918	31.578	47.569	29.629	382.101
3	26.493	36.992	47.185	28.975	387.286
4	26.518	30.033	46.634	27.679	387.646
5	29.202	30.335	47.545	28.109	382.766
6	26.461	34.882	47.716	29.064	383.555
7	25.103	30.708	48.155	29.918	383.829
8	26.283	30.550	47.521	29.791	386.223
9	25.421	31.484	46.545	27.728	384.958
10	26.195	30.340	48.200	27.825	387.074

Table 10.2.: Results for the four Queries and additionally the first one without joins

Third test on database with 840.134.615 datapoints

Execution	Query 1 wo Join	Query 1	Query 2	Query 3	Query 4
1	75.100	83.726	104.669	90.558	31449.329
2	75.366	82.864	119.347	91.445	32252.925
3	74.248	82.268	119.829	91.829	31363.360
4	75.357	81.038	105.461	89.119	31246.138
5	74.932	83.113	105.220	89.397	31258.628
6	75.818	81.508	104.354	92.112	31569.013
7	75.663	81.726	105.259	91.834	31739.085
8	77.018	81.363	105.945	91.197	31191.212
9	75.882	83.099	105.615	91.072	32097.022
10	74.921	81.414	104.882	90.302	31220.389

Table 10.3.: Results for the four Queries and additionally the first one without joins

List of Figures

List of Tables

Glossary

ACID Atomicity, Consistency, Isolation and Durability. 48

AIP Archival Information Package. 26

API Application Programming Interface. 12, 52, 79, 80

ArangoDB ArangoDB is a multi-model database system since it supports three data models (graphs, JSON documents, key/value) with a unified query language AQL (ArangoDB Query Language). 12

BJSON Binary JavaScript Object Notation. 11, 50, 51, 73

CCSDS Consultative Committee for Space Data Systems. 25

CEOS Committee on Earth Observation Satellites. 29

COSPAR Committee on Space Research. 25

DIP Dissemination Information package. 26

DMP Data Management Principles. 27, 28

D-SDA German Satellite Data Archive. 24

DL Data Lake. 14

DLR German Aerospace Center. xiv, 2, 17, 24, 33, 52

DSIG Data Stewardship Interest Group. 29

DWaaS Data Warehouse as a Service. 13

ESA European Space Agency. xiii, 1, 23, 25, 29, 44

ETL Extract, Transform, Load. 13, 14

GEO Group on Earth Observations. xiv, 26, 27, 28, 42

GPS Global Positioning System. 35

HALF Hefei Advanced Light Facility. 15

HDF5 Hierarchical Data Format 5. 79

HTML Hypertext Markup Language. 7

IBM International Business Machines. 1, 13

InfluxDB is a open source-source database, developed for time series data.. 15

InfluxQL Query Language of influxDB. 15

IPDA nternational Planetary Data Alliance. 23, 25

JSON JavaScript Object Notation. 7, 11, 12, 15, 50, 54

LDAP Lightweight Directory access protocol. 52

LTDP Long Time Data Preservation. xiv, 29, 40, 42, 44, 46, 47, 49, 51, 53, 55, 57, 59

MongoDB MongoDB is a source-available, cross-platform, document-oriented database program. Classified as a NoSQL database product, MongoDB utilizes JSON-like documents with optional schemas. 15

MORABA The mobile rocketbase, a department of the DLR. xiv, 2, 33, 34, 35, 37, 38, 39, 40, 41, 43, 45, 46, 55, 58, 72, 77, 78

MySQL free and open-source relational database management system emphasizing extensibility and SQL compliance. 7

NASA National Aeronautics and Space Administration. 1, 20, 23, 25, 33

NoSQL is an approach to database design that focuses on providing a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. xiii, 7, 9, 10, 11, 15, 18

OAIS Open Archival Information System. xiv, 18, 20, 25, 26, 41, 43, 84

OrientDB OrientDB is an open source NoSQL database management system written in Java supporting graph, document and object models. 12

PDF Portable Document Format. 7

PDS Planetary Data System. 1, 23

PostgreSQL free and open-source relational database management system emphasizing extensibility and SQL compliance. 7, 15, 16, 48, 49, 50, 51, 52, 54, 63, 64, 70, 73, 74, 84

PPDS Pilot Planetary Data System. 23

PSA Planetary Science Archive. xiii, 1, 23, 24, 44, 71

QDPS query datapoints per second. 16

RLS Row Level Security. 52, 53

SIP Submission Information Package. 26

SQL Structured Query Language. 7, 10, 11, 12, 15, 47, 54, 62, 63

TimescaleDB is a open-source extension for the open-source database postgreSQL. 15, 16, 48, 49, 62, 63, 64, 65, 70, 74

UI User Interface. 54

WGISS Working Group on Information Systems and Services. 29

XML JavaScript Object Notation. 7, 11

Bibliography

[ABD+10] Mirko Albani, Vincenzo Beruti, M. Duplaa, C. Giguere, C. Velarde, Eberhard Mikusch, M. Serra, Jens Klump, and Matthias Schroeder. *Long term preservation of earth observation space data - European LTDP Common Guidelines (Version 1.1)*. 01 2010.

[AM20] Mirko Albani and Iolanda Maggio. Long-term data preservation data lifecycle, standardisation process, implementation and lessons learned. *International Journal of Digital Curation*, 15(1):10–, 2020.

[AMMC23] M. Albani, K. Molch, I. Maggio, and R. Cosac. *Long Term Preservation of Earth Observation Space Data Preservation Guidelines*. 03 2023.

[Apo24] Apollo GraphQL. Apollo GraphQL: Data Graph Platform, 2024. `https://www.apollographql.com/` Accessed: 2024-10-30.

[Bar07] Sunita Barve. File formats in digital preservation. In *INTERNATIONAL CONFERENCE ON DIGITAL LIBRARIES, Bangalore*, 2007.

[BD84] H. Boral and D. DeWitt. A methodology for database system performance evaluation. pages 176–185, 1984.

[BVB+18] S. Besse, C. Vallat, M. Barthelemy, D. Coia, M. Costa, G. De Marchi, D. Fraga, E. Grotheer, D. Heather, T. Lim, S. Martinez, C. Arviset, I. Barbarisi, R. Docasal, A. Macfarlane, C. Rios, J. Saiz, and F. Vallejo. Esa's planetary science archive: Preserve and present reliable scientific data sets. *Planetary and Space Science*, 150:131–140, 2018. Enabling Open and Interoperable Access to Planetary Science and Heliophysics Databases and Tools.

[CLR23] Luca Clissa, Mario Lassnig, and Lorenzo Rinaldi. How big is big data? a comprehensive survey of data production, storage, and streaming in science and industry. *Frontiers in Big Data*, 6, 2023.

[CLZ+23] Huhang Chen, G.F. Liu, DaDi Zhang, Tian Qin, and X.K. Sun. A study of performance comparison of databases for half data archiving system. *Journal of Instrumentation*, 18, 2023.

[COM+09] Daniel Crichton, Pedro Osuna, Teresa Maria, Maria Capria, Reta Beebe, J. Hughes, Yukio Yamamoto, Jesus Salgado, and Yasumasa Kasaba. Developing the international planetary data alliance. 01 2009.

[DHK+85] S. Demurjian, D. K. Hsiao, D. Kerr, Robert C. Tekampe, and R. J. Watson. Performance measurement methodologies for database systems. pages 16–28, 1985.

[Dua24] Fabio Duarte. Amount of data created daily (2024), Jun 2024. https://explodingtopics.com/blog/data-generated-per-day#how-much Accessed: 2024-12-19.

[EI19] Bouchra El Idrissi. Long-term digital preservation: A preliminary study on software and format obsolescence. In *Proceedings of the ArabWIC 6th Annual International Conference Research Track*, ArabWIC 2019, New York, NY, USA, 2019. Association for Computing Machinery.

[Fow09] K. Fowler. The future of sensors and sensor networks survey results projecting the next 5 years. *2009 IEEE Sensors Applications Symposium*, pages 1–6, 2009.

[GEO24] GEO Data Working Group. GEO data management principles implementation guidelines, 2024. https://gkhub.earthobservations.org/records/v56md-kqg97. Accessed: 2024-10-23.

[GLVF21] Ignacio González Liaño and Marta Vázquez Fernández. Use of timescaledb as a database for ocean-meteorological data storage. *Instrumentation Viewpoint*, (21):40–41, 2021.

[Gor21] Eresh Gorantla. Postgres jsonb usage and performance analysis, 07-09-2021. https://medium.com/geekculture/postgres-jsonb-usage-and-performance-analysis-cdbd1242a018 Accessed: 2024-12-01.

[Gra07] Jim Gray. Data management: Past, present, and future. *CoRR*, abs/cs/0701156, 2007.

[Gra24] GraphQL Foundation. GraphQL: A Query Language for Your API, 2024. https://graphql.org/ Accessed: 2024-11-05.

[Gro23] PostgreSQL Global Development Group. *PostgreSQL: The World's Most Advanced Open Source Relational Database*, 2023. https://www.postgresql.org/ Accessed: 2024-10-29.

[GSB21] Darko Golec, Ivan Strugar, and Drago Belak. The benefits of enterprise data warehouse implementation in cloud vs. on-premises. In *Proceedings of the ENTRENOVA - ENTerprise REsearch InNOVAtion Conference, Hybrid Conference, Zagreb, Croatia, 9-10 September 2021*, pages 66–74, Zagreb, 2021. IRENET - Society for Advancing Innovation and Research in Economy.

[HP12] Bernd Holzhauer and Dr. Osvaldo Peinado. Protecting mission data against loss. In *SpaceOps2012*, Juni 2012.

[Inf24] InfluxData. InfluxData: Open Source Time Series Database Platform, 2024. https://www.influxdata.com/ Accessed: 2024-10-30.

[JPA+12] Nishtha Jatana, Sahil Puri, Mehak Ahuja, Ishita Kathuria, and Dishant Gosain. A survey and comparison of relational and non-relational database. *International journal of engineering research and technology*, 1, 2012.

[KMS+14] Stephan Kiemle, Katrin Molch, Stephan Schropp, Nicolas Weiland, and Eberhard Mikusch. Big data management in earth observation: the german satellite data archive at dlr. In P. Soille and P.G. Marchetti, editors, *Proceedings of the 2014 conference on Big Data from Space (BiDS'14)*, pages 46–49. Publications Office of the European Union, 2014.

[KV04] V. Koutsonikola and A. Vakali. Ldap: framework, practices, and trends. *IEEE Internet Computing*, 8(5):66–72, 2004.

[LH19] Jiaheng Lu and Irena Holubová. Multi-model databases: A new journey to handle the variety of data. *ACM Comput. Surv.*, 52(3), jun 2019.

[MAHK23] Aya Mohamed, Dagmar Auer, Daniel Hofer, and Josef Küng. A systematic literature review of authorization and access control requirements and current state of the art for different database models. *International Journal of Web Information Systems*, 20, 10 2023.

[MAP19] Bunjamin Memishi, Raja Appuswamy, and Marcus Paradies. Cold storage data archives: More than just a bunch of tapes. In *Proceedings of the 15th International Workshop on Data Management on New Hardware*, DaMoN'19, pages 1:1–1:7, New York, NY, USA, Juli 2019. ACM.

[Mas23] Joan Maso. Ogc cloud optimized geotiff standard. `http://www.opengis.net/doc/is/COG/1.0`, 2023. Submission Date: 2022-06-29, Approval Date: 2023-05-08, Publication Date: 2023-07-14.

[MFP+19] S. Martino, Luca Fiadone, A. Peron, Alberto Riccabone, and Vincenzo Norman Vitale. Industrial internet of things: Persistence for time series with nosql databases. *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 340–345, 2019.

[NM22] Athira Nambiar and Divyansh Mundra. An overview of data warehouse and data lake in modern enterprise data management. *Big Data and Cognitive Computing*, 6(4), 2022.

[OMST14] Justice Opara-Martins, Reza Sahandi, and Feng Tian. Critical review of vendor lock-in and its impact on adoption of cloud computing. In *International Conference on Information Society (i-Society 2014)*, pages 92–97, 2014.

[PETS12] Anna S Palaiologk, Anastasios A Economides, Heiko D Tjalsma, and Laurents B Sesink. An activity-based costing model for long-term preservation and dissemination of digital research data: the case of DANS. *Int. J. Digit. Libr.*, 12(4):195–214, September 2012.

[Pos24] PostgreSQL Global Development Group. *PostgreSQL Documentation: Row Security Policies*, 2024. `https://www.postgresql.org/docs/current/ddl-rowsecurity.html` Accessed: 2024-12-19.

[QnMFGRC23] Antonio Quiña Mera, Pablo Fernandez, José María García, and Antonio Ruiz-Cortés. Graphql: A systematic mapping study. *ACM Comput. Surv.*, 55(10), February 2023.

[Rea24] React Contributors. React: A JavaScript library for building user interfaces, 2024. `https://react.dev/` Accessed: 2024-10-30.

[RZ19] Franck Ravat and Yan Zhao. Data lakes: Trends and perspectives. In Sven Hartmann, Josef Küng, Sharma Chakravarthy, Gabriele Anderst-Kotsis, A Min Tjoa, and Ismail Khalil, editors, *Database and Expert Systems Applications*, pages 304–313, Cham, 2019. Springer International Publishing.

[Sha20] Baji Shaik. *PostgreSQL Configuration: Best Practices for Performance and Security*. Apress, Berkeley, CA, 2020.

[SPK+22] Andriy Sultanov, Maksym Protsyk, M. Kuzyshyn, Daria Omelkina, Vyacheslav Shevchuk, and Oleg Farenyuk. A statistics-based performance testing methodology: a case study for the i/o bound tasks. *2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT)*, pages 486–489, 2022.

[SPT19] Sirko Schindler, Marcus Paradies, and Andre Twele. Here is my query, where are my results? a search log analysis of the eoweb geoportal. In *2019 Conference on Big Data from Space (BiDS'19)*, 2019.

[SRMP02] Don Sawyer, Lou Reich, Patrick Mazal, and Nestor Peccia. The open archival information systems (oais) reference model and its usage. 2002.

[SS13] Seref Sagiroglu and Duygu Sinanc. Big data: A review. In *2013 International Conference on Collaboration Technologies and Systems (CTS)*, pages 42–47, 2013.

[SSSF09] Rolf Sint, Stephanie Stroka, Sebastian Schaffert, and Roland Ferstl. Combining unstructured, fully structured and semi-structured information in semantic wikis. 01 2009.

[Tal13] Paul P. Tallon. Corporate governance of big data: Perspectives on value, risk, and cost. *Computer*, 46(6):32–38, 2013.

[Tim23] Inc. Timescale. *TimescaleDB: An Open Source Time-Series Database*, 2023. `https://www.timescale.com/` Accessed: 2024-10-29.

[VCL23] T. Verner-Carlsson and V. Lomanto. *A Comparative Analysis of Database Management Systems for Time Series Data*. Dissertation, 2023.

[VS14] Phillip Viana and Liria Sato. A proposal for a reference architecture for long-term archiving, preservation, and retrieval of big data. In *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 622–629, 2014.

[Wor24] World Meteorological Organization. Group on earth observations (geo), 2024. `https://wmo.int/activities/group-earth-observations-geo`.