Review

The Challenges of Reproducibility for Research Based on Geodata Web Services

Massimiliano Cannata *, Maxime Collombin, Olivier Ertz, Gregory Giuliani, Jens Ingensand, Claudio Primerano, Daniele Strigaro

Posted Date: 29 December 2023

doi: 10.20944/preprints202312.2316.v1

Keywords: open science; geospatial web services; reproducibility; FAIR; interoperability; reproducible research;

Review

The Challenges of Reproducibility for Research Based on Geodata Web Services

Cannata Massimiliano [1,*], Collombin Maxime [2], Ertz Olivier [2], Giuliani Gregory [3], Ingensand Jens [2], Primerano Claudio [1] and Strigaro Daniele [1]

[1] University of Applied Sciences Southern Switzerland (SUPSI), Mendrisio, Switzerland
[2] School of Engineering and Management Vaud, HES-SO, University of Applied Sciences and Arts Western Switzerland
[3] University of Geneva, Geneva, Switzerland
* Correspondence: massimiliano.cannata@supsi.ch

Abstract: Modern research applies the Open Science approach that fosters the production and sharing of Open Data according to the FAIR (Findable, Accessible, Interoperable, Reusable) principles. In the geospatial context this is generally achieved through the setup of OGC Web services that implements open standards that satisfies the FAIR requirements. Nevertheless, the requirement of Findability is not fully satisfied by those services since there's no use of persistent identifiers and no guarantee that the same dataset used for a study can be immutably accessed in a later period: a fact that hinders the replicability of research. This is particularly true in recent years where data-driven research and technological advances have boosted frequent updates of datasets. Here, we review needs and practices, supported by some real case examples, on frequent data or metadata updates in geo-datasets of different data types. Additionally we assess the currently available tools that support data versioning for databases, files and log-structured tables. Finally we discuss challenges and opportunities to enable geospatial web services that are fully FAIR: a fact that would provide, due to the massive use and increasing availability of geospatial data, a great push toward open science compliance with ultimately impacts on the science transparency and credibility.

Keywords: open science; geospatial web services; reproducibility; FAIR; interoperability; reproducible research

1. Introduction

This article discusses the reproducibility of researches that have been based on datasets offered by interoperable open geospatial Web services and that are subject of frequent modifications. We explore the current context and a few cases of frequent data changes of different geospatial data types and discuss some available technological solutions to support data versioning. The emphasis of this paper is on challenges and needs of practical solutions.

1.1. Open Science and Open Research Data

Digitalization and collaborative approach are the essential aspects driving Open Science, which is the modern way of conducting research. As discussed by Ramachandran [1] Open Science in its broader denotation can be defined as "a collaborative culture enabled by technology that empowers the open sharing of data, information, and knowledge within the scientific community and the wider public to accelerate scientific research and understanding". Toward this overarching goal, three major objectives have been identified: (1) increase the accessibility to the scientific body of knowledge, (2) increase the efficiency of the processes to share research outputs and findings, and (3) improve the evaluation of the science impact considering new metrics. Due to technological advances of the last decades, modern research is, today, mainly data-driven [2–4]. For this reason Open Research Data (ORD) is extremely important, it refers to "the data underpinning scientific research results that has no restrictions on its access, enabling anyone to access it." [5]. With means to openly share data, the intent is to accelerate and boost new findings and innovations. It minimizes data

 doi:10.20944/preprints202312.2316.v1

duplications and enables interdisciplinary and wider collaborative research. To be effectively used by other researchers, ORD need to be shared following specific principles of Findability, Accessibility, Interoperability, and Reuse (FAIR) (Wilkinson et al., 2016) and this led to the creation of several data services that permits to register, store, find and access data following interoperable metadata standards. The mostly used data services, like for example Zenodo [6], PANDORA [7], ARCHE [8] or UK Data Archive [9], offer free data repositories which are registered on the Registry of Research Data Repositories (re3data) [10]. In most of the cases, they adhere to ORD best practices by offering open data access, associating a license to data, making them persistent, providing unique citable identifiers (DOI), adopting repository standards, and providing a defined data policy. Nevertheless, in most of those repositories it is only possible to deposit static files, preferably archived using an open standard format (i.e. CSV, ODT, JSON) and associated with standard metadata. However, to fully exploit ORD with modern applications using for example machine learning techniques, big data requires specialized services that offer a systematic and regular delivery of Analysis Ready Data (ARD) and preprocessing/filtering capabilities [11]. Sharing ARD perfectly fit with the European vision of establishing Data Spaces as an interoperable digital place to facilitate data exchange and usage in a secured and controlled environment among different disciplines with the goal of boosting innovation, economic growth and digital transformation [12]. This concept goes beyond the simple technical data sharing issues and encompasses the need of offering a space to share data that is compliant with privacy and security regulation.

1.2. Geospatial Data services

In the geospatial context, operational data sharing has been implemented by means of European, national and regional Spatial Data Infrastructures (SDIs). They have been implemented based on the same sharing principles which, from a technical point of view, led to the adoption of interoperable geoservices by which today thousands of geospatial layers are offered to millions of applications worldwide. Those adopted interoperable geostandards are mainly the results of the efforts of the Open Geospatial Consortium (OGC) which is a cross-domain Standards Developing Organization (SDO) composed of governments, academia and commercial partners. More recently, in 2017, the OGC and the World Wide Web Consortium (W3C), which sets standards for the Web, joined the forces to produce a reference document providing indication on best practices for publishing spatial data on the Web [13]. As a consequence OGC has begun to develop a new family of standards: the OGC APIs [14]. These standards, built on the legacy of previously defined OGC Web Service standards (WMS, WFS, WCS, WPS, etc.), aim at facilitating the integration of spatial data with other information by defining resource-centric APIs that take advantage of modern web development techniques. Unlike the OGC Web Service Standards, the OGC APIs are based on a REST API rather than the SOAP protocol. They favor JSON encoding over XML. They also come with documentation based on the OpenAPI specification to facilitate their discovery and integration, as well as HTML rendering and endpoints in JSON-LD (JavaScript Object Notation for Linked Data) to facilitate their indexing by search engines. These standards are also designed as "Building Blocks" that can be used to assemble new APIs for accessing geospatial content on the web. The modules are defined not only by the requirements of the standards specified in the OGC Standards Programme, but also by prototyping and interoperability testing as part of the OGC Collaborative Solutions and Innovation Programme.

1.3. Time varying data

The technological growth in the last decades led to the explosive increment of time-varying data which dynamically change to represent phenomena that grows, persists and decline [15], or that constantly vary due to data curation processes that periodically insert, update, or delete information related to data and metadata. As discussed by Saracco [16], dealing with dynamic data has consistently played a pivotal role in facilitating a wide range of analyses, such as examining changes in client accounts for financial institution audits, assessing the clinical progression of patients for legal proceedings, evaluating insurance policy terms at the time of accidents, identifying disparities in

 doi:10.20944/preprints202312.2316.v1

travel itineraries involving car and hotel reservations, and adjusting interest rates for banks upon detecting errors. In addressing such cases two different concepts of time can be identified: *system-time* and *business-time* [16]. While *business-time* can be defined as the instant/period for which the data is meant to be used (often referred to as "valid-time" or "application-time"), *system-time* relates to the data state as stored in a specific instant/period (often also referred to as "transaction-time"). So for example, in the case of an application tracking the position of a delivery track and the corresponding air temperature the system would record the coordinates and the air degrees Celsius. After a month, if the track undergoes a revision in which the temperature sensor was found to have a bias of 0.002 degrees and the raw GPS data were manually collected and reprocessed for improved position accuracy, data would be updated accordingly. Answering the questions "where was the track on August 30 at 11:30? What is the average temperature in May on London Street? What is the distance covered by the track in 2023?" requires the usage of the *business-time*. Answering questions "Which data did we use to compile the delivery report on August 25 at 17:15? How did the position of the track change after GPS data processing?" requires the usage of *system-time*. From a logical perspective *business-time* is characterized to be maintained by the user, future dates and times may make complete sense, its resolution (day, month, microseconds, etc.) is decided by the application. On the contrary, *system-time* is managed automatically, future dates are not permitted and its resolution should be the finest possible.

2. Reproducibility research in the geospatial sector

Based on the current trends and data availability, the ability to link Open Science concepts with interoperability and time-varying data management is paramount. In particular, the capability of obtaining results consistent with a prior study using the same materials, procedures, and conditions of analyses is very important since it increases scientific transparency, fosters a better understanding of the study, produces an increased impact of the research and ultimately reinforces the credibility of science [17,18]. In the Open Science paradigm this is indicated as Reproducible Research, and it can be guaranteed only if the same source code, dataset, and configuration used in the study is available. For geospatial data, while the presented OGC standards enable an almost FAIR [19] and modern data sharing, they do not adequately support the reproducibility concept as pursued in Open Science. In other words, they do not offer any guarantee that the geodata accessed in a given instant can be persistently accessed, immutably, in the future. In fact, while business-time is considered in several aspects and standards like Part 3 from OGC API - Features including temporal filtering [20], Sensor Observation Service [21] or OGC API Moving Features [22], at the best knowledge of the authors, none of them support the *system-time*. Releasing different dataset versions (e.g. storing exported datasets in FAIR repositories or offering a layer referencing a fixed time instant) can be a solution. Nevertheless, in many cases, where data (and metadata) are frequently updated and datasets have large size, this is not efficient nor applicable. Additionally, this approach may hinder the seamless capacity of analyses of data variations.

This is confirmed by Nüst and Pebesma [23] that produced a comprehensive summary of the state of the art in reproducible research within the geospatial domain. In the manuscript they recognize that only a small body of work on reproducibility in the geospatial domain was available. They underlined that reproducibility might be achieved only when physical, logical and cultural components are available and identified that the main challenge is the general poor knowledge of reproducibility practices by researchers. Other barriers that they highlighted were related to:

1. The utilization of proprietary software which is often subject to licensing restrictions prevents reproduction.
2. The multitude of tools frequently employed in a single geospatial research project poses a challenge to replicability.
3. The reliance on geospatial infrastructure that depends on online services can lead to obstacles in accessing the original dataset due to potential changes.
4. Analyzing extensive datasets is often executed in proximity to the data source using online services, which would necessitate open accessibility to the server implementations.

 doi:10.20944/preprints202312.2316.v1

5. While free platforms provide scripting capabilities for data processing, the environment is subject to change and, as a result, may not ensure reproducibility.

Cerutti et al. [24] in their study, which replicated and compared three studies conducted on disaster response using different geospatial algorithms, proposed the use of an analytics platform [25] which includes geospatial functions to create scientific workflows and enhance reproducibility. Similarly, registering the analysis workflow in computational notebooks enables the ordered re-execution of processing steps: this approach has been adopted for example by the GRASS GIS community (GRASS GIS Jupyter notebooks) and ESRI (ArcGIS notebooks). Nevertheless, the reproducibility of workflows does not guarantee that the code is executed using the same data and environment used during the study. For this reason, Yin et al. [26] presented a cloud-based solution, named CyberGIS-Jupyter, that combines Jupyter notebook with docker technology to support computational reproducibility and scalability of geospatial analyses. While Jupyter notebook guarantees the reproducibility of the workflow, the docker technology permits to reproduce the environment, with exactly the same software and libraries versions, where the geospatial processes were executed. Kedron and Li [17] highlighted in his review that even if the computational reproducibility is guaranteed, it does not include itself two essential components for the full reproducibility of research: the record of task coordination and of conceptual decision-making. For this reason, research notebooks and research management software like the Open Science Framework [27] have been used to capture research provenance (steps and decision criteria) in addition to processing workflow including approaches and pre-analysis plans. The reported approaches present solutions that contribute to solving several reproducibility challenges, nevertheless they do not address the issue of accessing the original dataset when online services from a geospatial infrastructure are at the base of a study.

With the aim of understanding how system-time, and reproducibility as a consequence, could be addressed and managed in geospatial data services, this work aims at identifying the requirements, challenges and opportunities by reviewing: (1) how the geospatial domain is addressing reproducibility, (2) how updates of time varying spatial data happens in real case applications, and (3) which technological solutions exists to manage *system-time* for big-data spatio-temporal datasets.

It may be worth mentioning that the OGC issued a Call for participation (OGC, 2023b) for the Open Science Persistent Demonstrator (OSPD) Initiative. Although important for the replicability of science, this initiative focuses on platforms and workflows to demonstrate how a federation of OGC based services can offer FAIR Open Data in support of Open Research. Nevertheless, the accessibility of data available at a specific time is not directly addressed and might be somewhat beyond the scope of this research project.

The remainder of the paper presents the results of the literature review for each of these three aspects and finally presents a discussion highlighting possible solutions and approaches to be investigated in the future.

3. Updates and Changes of Time Varying Geospatial Data

In this section we evaluated the needs and practices supported by some real case examples related to common operations that update data or metadata of the different geospatial data types, specifically sensor observations, vector datasets and raster series.

3.1. Updates and Changes in Time of Data from Sensor Observations Services

In the geospatial domain, sensor observations are mainly addressed by two standard web services specifications from the OGC: the Sensor Observation Service (SOS) and the Sensor Things API (STA). While SOS is based on the Simple Object Access Protocol (SOAP) and data are encoded in Extensible Language Markup (XML) the STA is based on the use of RESTful services following the OData's specification [28] and data are encoded in JSON format. Both specifications offer access to sensor data and metadata along with transactional capabilities. Apart from some specific differences in the data model [29] and requests the two standards can be from a logical point of view comparable.

 doi:10.20944/preprints202312.2316.v1

Based on the previously reported challenges of reproducibility in the geospatial sector these standards are exposed to potential changes of datasets in time, in fact an authorized user can change data using the offered transactional features. Best practices followed to serve ARD from sensor networks very often include post-processing (after the original acquisition from the sensor) to reduce uncertainties in further analyses which produce knowledge and wisdom. As discussed by Krishnamurthi et al. [30] these processing include: (i) denoising to eliminate most of the noise signals in data, (ii) missing data imputation to deal with incomplete data that are not supported by several analyses techniques (e.g. ML models), (iii) data outlier detection to identify data which has been incorrectly sensed due to external unpredictable factors, (iv) data aggregation of heterogeneous observations (difference in time and property) to reduce data transmission size and complexity. Additionally, after the processing phase data may require (v) data fusion which integrates multiple data sources to improve accuracy. At this point data are available for analyses as ARD.

Strigaro et al. [31] described a system collecting high frequency data of ecological and physical parameters from buoys that are located in lakes with the scope of monitoring and assessing lake ecosystem quality. A semi-automatic data quality control process was set to ensure consistency, reduce human bias and cope with large data flow. The process foresaw the testing of the data versus an ordered list of mathematical or logical criteria as proposed by the WMO guidelines on quality control procedures [32]. Tests are applied in real-time at the edge and in post-processing at the server side. At the edge, simpler data quality checks processes are run to aggregate raw data based on their quality, then aggregated data are transmitted to the servers. Then data are stored and a second quality control process applies a sequence of quality checks: when a check fails, observed values are flagged with the quality code referencing the not passed test. Finally, data that are suspicious, due to test failures, are manually evaluated and eventually corrected in the data storage.

Reported cases of time series data management shows that data preprocessing and quality assurance procedures modify data and metadata values as a consequence of real-time analyses or post-processing elaborations or lately manual correction. It is particularly interesting that the WMO [33] in its climate data management system specification includes as a required policy the “ability to reproduce specific data that were held in the climate database at a particular point in time” (policy 3.1.6.2 Data lineage traceability). Additionally, the document states: “It is becoming increasingly apparent that organizations will need to retain observations at multiple levels of quality from the raw observation through various edit and analysis processes in order to demonstrate the true lineage of a record and explain and justify the changes made to the raw observations.”

To better understand the entity of these changes in a real case study we have analyzed the transactional operations that have been executed on the hydro-met monitoring system in the Canton Ticino, Southern Switzerland, described in detail in Pozzoni et al. [34]. The monitored data are managed within a SOS compliant service based on the istSOS software [35]. The datasets, at the time of writing, include observations ranging from the 1978 to the 2023 and related to 298 sensors observing different properties related to rivers (e.g. flows and temperature) and meteorology (e.g. precipitation, temperatures and humidity). From 2015, year of the activation of a transactional log features offered by istSOS and that permits to register transactional operations executed on the service, we can note that out of 117,037,540 observations there have been 15,069,970 updates of single observation data or metadata, value, which, not considering multiple updates for the same values, correspond to about the 13% of the data. Percentage that reaches 23%, if we consider only the measures related to river height and precipitation, which are the measures that actually undergo a systematic quality control process (6,158,284 out of 26,820,857 observations). These percentages highlight how important it could be to access specific dataset status in time. Additionally, since every year an annual hydrobook is produced in the following year, data undergo a specific process of re-analyses where, for example, stage-discharge relationship curves are updated or data gap filled and the entire previous year aggregated data is re-calculated. As an example, in October 2023 hydrologists requested to re-calculates the daily aggregated data for all the discharge stations for the entire 2022, for a specific station for the entire 2021 and for another station for one week due to configuration changes.

 doi:10.20944/preprints202312.2316.v1

3.2. Updates and Changes in Time of data from Feature Services

Vector layers which vary in time due to continuous data additions and updates are generally managed with database or file sources and are very often offered by means of standard OGC WMS or WFS and more recently via the OGC API Feature. As an example of geospatial datasets undergoing a continuous update we can cite the cadastral data and the water protection zones. They are continuously updated to reflect changes that occur on land ownership or water policies. In particular, the groundwater protection zones [36] identify different areas (S1, S2, S3) centered around groundwater collections or recharge plants. Those areas regulate and limit the use and activities permitted so that potential pollution can be prevented and there is enough time to take protection actions in case of accidents. When new wells are established and/or new hydrogeological studies are conducted new protection zones are inserted or their variation is edited and, after a validation process, are published as WMS on the official portal on the Cadastre of public-law restrictions on land ownership of Canton Ticino (https://crdpp.geo.ti.ch/). Here the current situation is duly represented but there is no option to navigate and check how restrictions were represented in the past, which may be very relevant in case of disputes. Similarly, the cadastre of Canton Ticino is publicly available as WFS service (https://wfs.geo.ti.ch). This is the current version and while we can find in the attributes the date a feature entered in law, it does not offer any option to navigate in time the evolution of properties map.

A very popular geospatial vector dataset that is continuously updated is the OpenStreetMap (OSM) [37]. It is a crowdsourced geographic dataset started in 2004 and distributed under an open access license (Open Database License, ODbL) that permits free usage of the information collected by the more than 10 million registered users [38]. OSM maintains timestamped historic changesets that are a group of modifications set by a single user in a short period. It is possible to access weekly snapshots and changesets of the full dataset or the full history by downloading them in XML or Planet PBF files (https://planet.openstreetmap.org/). To analyze the creation process of OSM data some software were implemented by the are limited to examine and view a small portion of the database [39]. Martini et al. [40] presented a methodology for analysis of changes in OSM. In this paper the authors analyzed the changed objects in the city area of Karlsruhe, Germany. The produced maps show areas with up to 2 thousand new objects per year and up to 13 thousand objects updates per year. These numbers and the full history file with its size of more than 200 Gb provide a clear order of magnitude of the high variability of the data in time due to new features, deletion or modification of either geometric or semantic information.

3.3. Updates and Changes in Time of Raster Services

Concerning raster data, the principal source of data is represented by satellite Earth Observations (EO) data. Indeed, since the 70s and the launch of the first Landsat satellite, planet Earth is under continuous observation from many different types of satellites (e.g., optical, radar, hyperspectral) [11]. These satellites produce a continuous and increasing stream of observations from space. Among the different benefits of using satellite imagery for environmental monitoring, Merodio Gómez et al. [41] listed (1) *Temporal resolution*: capacity to capture data at different frequency of revisit (e.g., 5 days for Sentinel-2, 16 days for Landsat, up to 2.9 days in combined use mode and closer to the equator); and (2) *Time-series*: capacity to provide continuous data starting as early as 1972 (e.g., Landsat) as two important aspects [19].

Since the advent of open data license on Landsat data and the subsequent opening of the archive, it has allowed the move from diachronic (Before/After) analysis to near-real time analysis [42]. In particular, to tackle the big data challenges related to EO data handling [43], the emergence of EO Data Cubes allowed to efficiently and effectively manage and analyze large amounts of EO data [44,45], enabling spatio-temporal analysis of Analysis Ready Data (ARD) [46]. However, interoperability of Data Cubes is still a significant challenge [47], different existing and emerging standards can help deliver and leverage the power of EO data building, efficient discovery, access and processing services [48].

 doi:10.20944/preprints202312.2316.v1

OGC Web Map Service (WMS) and Web Coverage Service (WCS) are common standards that already have demonstrated their ability to handle satellites for visualization and download purposes. While both standards have a *time* parameter that can be used to extract a specific time-slice, it remains limited to that single operation. It cannot do operations on a time interval or nearest values. Another restriction, related to the semantic of the time parameter, is ambiguous that could refer to acquisition - processing or publication time. The OGC WMS Earth Observation profile recommends using the time parameter only for the acquisition time [49]. Giuliani et al. (2019) discuss in detail and demonstrate potential ways to properly handle the time dimension on existing OGC standards.

Among the new standards that have emerged in recent years, the Spatio-Temporal Assets Catalog (STAC - https://stacspec.org/en) provides a common structure for describing and cataloging spatiotemporal assets [50]. It tackles most of the issues previously mentioned and facilitates the creation of flexible spatio-temporal analysis workflows, removing the burden of creating specific pipelines for each different data collection one consumes. In conjunction with the use of Cloud Optimized Geotiff (COG - https://www.cogeo.org/) data format, time-dimension of raster array data is greatly facilitated.

To exemplify the benefits of this new specification, we looked at how EO Data Cubes have evolved. Before the emergence of STAC and COG, most Data Cubes, were ingesting data into their own premises, contributing to data duplication and redundancy leading potentially to differences in content and therefore in its temporal dimension (e.g., missing scenes, etc…) and at the end of time-series of data and/or products that could be different. With STAC it is now possible to easily index the reference source of data, therefore having one single entry point, and then facilitating the building of a consistent and up-to-date data repository of satellite imagery and related products. A good example is the Digital Earth Africa STAC end-point (https://explorer.digitalearth.africa/products) that gives access to time-series of Landsat 5-7-8-9 and Sentinel-2 imagery as well as derived products such as land cover, crop maps and observations of water, in a consistent way while efficiently enabling the handling of the time dimension by interactively querying the data repository like in the example of Landsat 8 Surface Reflectance data: https://explorer.digitalearth.africa/products/ls8_sr .

Nevertheless, existing and emerging standards are not properly handling backward compatibility of raster-based products (e.g., guarantee that I can access the data as they were yesterday... not like they are today). Indeed, for example in data cubes, if one reprocessed a given product (e.g snow cover) or ARD satellite imagery, with an improved version of an algorithm, then the only solution for versioning them is to create a new data collection with a different version number that can then be queried/accessed through an OGC-compliant API or Web service. A possible solution to be investigated is the use of the transactional WCS (WCS-T) that could partially solve this issue allowing one to automatically update other data cubes that have the same product with new time slices or completely new products [47].

4. Tools for data versioning

Traditionally data management tools maintain current valid data, so irreversible changes occur when the dataset is varying due to data acquisition (the dataset expands including new features) or data management (the dataset changes due to processing). Reproducible research, among the other requirements, compel the capacity to set up the same environment and software used in the study and to access used datasets in exactly the same state they were while running the experiment. As a consequence it is paramount that data storage and management tools offer data versioning features so that it is possible to access previously used versions.

Today, several applications, which benefit from big data, take advantage of accessing historical changes of data. In particular, these solutions have been pushed by Machine Learning (ML) models that are continuously calibrated using new or enriched data and therefore required to record how parameters changed along with the used dataset [51]. Since both science and industry are more and more data driven and the data production rate has quickly grown, data managers start moving from databases to object storage technology following the Data Lakes [52]. For this reason, most of the available solutions that offer data versioning capabilities relate to object storage (dataset version

 doi:10.20944/preprints202312.2316.v1

control) rather than databases (data versioning). As a consequence, today different tools exist to support the versioning of big data in a collaborative and open approach. Most of them take inspiration (and use) the base concept of Git which permits to collaboratively manage software code tracking changes to files executed by different people in time and allowing to back up old versions [53]. In the next paragraphs a short description of the key working principles of some of the adopted/available solutions for different approaches are presented. In spite of the Open Science principles, Open Source software solutions have been prioritized in this presentation. It has to be noted that not always scientific papers describing these tools are available and for this reason often generic web resources like blogs, websites, message chats or software documentation is used as a reference.

4.1. Data versioning of databases

Tracking the historical evolution of records or database version control mechanisms are generally based on the definition of a Slowly Changing Dimension (SCD) [54] which is a dimension that registers, and permits to manage, the evolution in time of values in a database table. Three types of SDCs exist: *(i) type 1* stores the latest valid values of a record and that is the standard database rule, *(ii) type 2* stores all the versions of the record registering the period for which that values was active, and *(iii) type 3* stores the current and previous values only of a record. In December 2011, ISO/IEC published an updated version of the SQL standard, SQL:2011 [55] which introduced the capability of managing temporal tables. This includes the support of time period data type which can be declared as primary or foreign key, and support a number of filtering operations (i.e. overlaps, equals, contains, precedes, succeeds, immediately precedes, immediately succeeds). A period column can have any name except SYSTEM_TIME, which is a reserved name to enable system-time features as SDC type 2. Several databases implemented the SQL:2011 specification as described by Jungwirth [56]. In addition to relational databases, Soroush and Balazinska [57] presented a methodology for extending column stores (array databases) with versioning.

MariaDB - In MariaDB you can enable the system-time versioning of a table using the syntax "WITH SYSTEM VERSIONING" which adds the "ROW_START" and "ROW_TO" pseudo-columns that do not appear in SELECT statements and are populated automatically. The ROW_START is populated with the insertion timestamp while ROW_END with an instant far in the future if the record is valid or the instant the row has been updated/deleted. Filtering using the system-time can be performed using the "AS OF" to extract a specific version of the data in a specific instant or the "FROM … TO" or "BETWEEN" statements to extract the record as they were valid in a provided period. Versioned tables with system-time can be partitioned so that historical rows and current valid rows are separated, optionally users can set the historical partition to be partitioned every n records [58].

IBM DB2 - System-time is implemented similarly to MariaDB with a few syntax differences, like naming "ROW BEGIN/END" instead of START/TO. System-time pseudo-columns columns are not accessible in SELECT statements and filtering does not support BETWEEN statements [59].

Oracle - Oracle has its own implementation of historical values that do not comply with SQL:2011. It allows you to declare a PERIOD but not as a primary key and periods can have null values. Oracle Database supports Flashback Time Travel feature implementing SDC type 2 that can be configured with a retention time on a tablespace or identified tables making it easy to undo or query historical stored values [60,61].

MS SQL Server - System-time tables are supported but with non fully standard syntax "WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.this))" where historical records are saved in an invisible table that can be named and queried as any normal table. Filtering supports standard options like *MariaDB* [62].

PostgreSQL - Postgresql does not support system-time. There are a few projects implementing such a feature but they are not official extensions and are under development. A few github repositories implement a solution in Pl/PgSQL [63,64]. An extension dated 2018 is proposed on the

 doi:10.20944/preprints202312.2316.v1

PostgreSQL Extension Network (PGXN) website. None of these are officially supported by postgreSQL.

OrpheusDB (http://orpheus-db.github.io/) - OrpheusDB is a layer installed on top of relational databases and expose git-like command [65]. It stores data in tables following the SDC type 2. In OrpheusDB records are immutable and are archived in Collaborative Versioned Dataset (CVD) recording *record id* (rid) and *version id* (vid) with other associated metadata like *creation time*, *commit time*, *committer* and a *commit message*. OrpheusDB implements a CLI (Command Line Interface) with, among other, checkout and commit commands. SQL can be performed on a version with the run command that takes the SQL as an input, translates and executes it against the database for a specific version ("SELECT …. FROM version X OF CVD WHERE …."). In his paper Huang et al. [66] demonstrated the solution on postgreSQL and presented a graphical interface to navigate through the version tree.

Dolt (https://docs.dolthub.com/) - While not found in scientific literature, Dolt is a MySQL git-like database versioning system. According to its documentation Dolt treats tables as files and registers modifications in a stage area (so called "working set") which is the current database version used when queries are executed. When a Dolt commit is performed a new version is persisted so that differences between versions, and metadata can be explored. Optionally, it is possible to configure Dolt so that at each SQL commit a dolt commit is executed but according to Sehn [67] in this case you lose the capability of annotating commits with messages and the complexity of the commit graph could become hard to be used. Dolt supports branches, diffs and merges.

4.2. Data versioning of files

When data are not structured in relational databases but managed in files, changes can be recorded using Git. Unfortunately Git has not been designed to manage large datasets, in fact it extracts the list of changes (*diffs*) from stored file snapshots, a fact that limits its performance [68]. For this reason some solutions have been implemented to overcome this issue and extend Git to support large files.

Git Large File Storage (Git LFS, https://git-lfs.com) - Kandpal [69] proposed a tool for collaborative development of machine learning models, based on the Git Large File Storage and described in detail Git LFS functioning and limitations. Its main feature is that it has been designed as a Git extension that permits tracking large binary files seamlessly in Git. It works similarly to ordinary Git solutions allowing users to add, commit, push, fetch and checkout file modifications, but instead of storing binary files in Git it replaces them with a text pointer to an external resource that hosts the actual file. When a file is tracked it is managed as a single object thus any modification of the file creates a new copy of the entire object in the storage. For this reason, its drawback is that storage size is proportional to the commits regardless of the size of the modification. Additionally, it is not possible to get meaningful *diffs* between versions but only get acknowledged that files are, or not, bitwise identical.

Data Version Control (DVC, https://dvc.org/) - As discussed by Peuster et al. [70] Data Version Control (DVC) is a software specifically implemented to facilitate management of Machine Learning models and data in Git fashion, using external storages to store binary files and Git as a reference. According to the DVC online documentation (https://dvc.org/doc/user-guide) DVC differs from Git-LFS mainly because it doesn't require specific servers but can use any cloud storage solution. Not much can be found on the mechanism for data versioning on the project documentation, but thanks to an answer from the co-founder of DVC on stack overflow (https://stackoverflow.com/a/60366262) we understand that files are tracked as single objects and thus replicated in case of any part modification.

Lake FS (https://lakefs.io/) - Lake FS is yet another version control system based on the Git approach that allows managing files stored in cloud storages. Like DVC its primary objective is to record Machine Learning models with its associated training dataset. Lake FS permits to branch, commit and merge data which could scale to petabytes allowing to manage data across different cloud storages. It can also revert changes in data. According to Keun-Tae [71] it has been created specifically to improve performance on scalable systems.

 doi:10.20944/preprints202312.2316.v1

Pachyderm (https://www.pachyderm.com) - Pachyderm, is an open-source platform for managing data pipelines and the associated input/output data. It manages data versioning and lineage by using a combination of technologies: it leverages Git to manage version control using distributed file systems like Hadoop or S3 to store large datasets in addition to databases or key-value stores to record information on how data is generated, transformed, and consumed within the system. When data are committed file hash is produced and file recorded in data storage. When changes are committed it records the variations between the previous version and the new version so that any particular state of data can be then identified by commits [72]. The usage of docker technology then allows to encapsulate data processing steps and create portable, reproducible data pipelines.

Kart (https://docs.kartproject.org) - Kart is a distributed version-control built on Git specifically implemented for handling geospatial and tabular data. No scientific papers could be found on the software, but according to its documentation it supports different geospatial data types including raster, point cloud and vector datasets. In case of rasters or point-could due to the size of the data Kart uses Git LFS. Specific datasets are stored using defined data formats and folder structures in git, so for example rasters are stored as GeoTIFF in the folder *.raster-dataset.v1* and point clouds are stored as LAZ files in the folder *.point-cloud-dataset.v1* (using Git LFS), both have a nested structure with two folders: *meta* for the metadata and *title* with the actual data (stored in Git). Similarly, for vector and table data type Kart uses a *.table-dataset* folder with *meta* and *feature* subfolders storing all the information in Git. Vectors/tables data are stored as a single file per feature/row therefore modifications are versioned at row levels which permit, by using the metadata, to reconstruct a dataset at a specific commit.

4.3. Data versioning of Log-Structured Tables

Modern columnar data formats like Apache Parquet [73] and ORC [74] due to their characteristics of being optimized for storage and retrieval, they became very popular. Nevertheless, their characteristics of being immutable pose a limitation in their adoption in all the cases where frequent updates are required. To overcome this issue, while keeping the benefit of those formats, the Log-Structured Tables (LSTs) solution has been implemented. It adds on top of the immutable columnar data formats a metadata layer that records the versioning of tables and parameters to enable the interaction through the processing engine [75]. This solution is shifting the paradigm of data storage from traditional data warehouse due to its capability of offering ACID transactions and supporting frequent table modification by creating a new immutable columnar file containing the changes. Additionally, it makes use of distributed cloud storage systems, and therefore with respect to traditional data warehouses, is simple and fast to scale. Nevertheless if modifications are too numerous, the metadata overly may slow down the process of data querying and retrieval. To overcome this issue several approaches have been adopted by different solutions. Popular LST solutions are herein reported.

Delta Lake (https://delta.io) - Delta Lake makes use of a transaction log along with Apache Parquet files to offer ACID properties over cloud object storages so that consistency and reliability are offered. It has been used to store Online Transaction Processing (OLTP) data, time series and logs. For querying data a fast query engine for lakehouse systems like Photon [76] can be used for the integration of SQL operation.

Apache iceberg (https://iceberg.apache.org/) - Apache Iceberg is a LSTs format that has been designed for high performance and that connects with engines like Spark, Trino, Flink, Presto and object storages. This combined solution supports full SQL, schema evolution, time travel and optimization. It adds metadata layers to the existing files and exposes them as iceberg tables to the engines while maintaining traditional database features like ACID transaction and time travel. Every table change requires that the associated metadata file is replaced by a new one. The format requires that the data are immutable (not changed or moved after they are written), the files support seek and can be deleted or, if maintained, marked as deleted so that the capability of time travel is exposed. At the time of writing there is no official support for geospatial support despite some efforts from the community has identified [77]. In this thread a geospatial solution based on Apache iceberg and name

 doi:10.20944/preprints202312.2316.v1

of geoLake (https://github.com/spatialx-project/geolake) has been mentioned. Nevertheless, currently it has no issue recorded and is not clear how the project will continue (no roadmap) and support the evolution of Apache Iceberg in the future.

Apache Hudi (https://hudi.apache.org/) - Apache Hudi (Hadoop Upserts Deleted Incrementals), like Apache Iceberg has been create to support large data storage in distributed storage systems. It stores tables in folders and subfolders which comprise file groups sliced in partitions which ultimately contain data in parquet format. Depending on the configuration, changes on tables can be managed with the copy-on-write or merge-on-read approaches: the first creates a copy of the parquet file on any changes and is optimized for read-intensive cases, the second store the updates in delta files that are then merged when the data are requested and is indicated for write-intensive situations [78].

5. Research Challenges and Opportunities

OGC open standards have been widely adopted in the geospatial sector to create Spatial Data Infrastructures that provide a massive amount of geospatial data with interoperable services and formats and that researchers have used to conduct various researches in almost all scientific disciplines. This generated a great push toward FAIR data sharing by offering findable, accessible and interoperable infrastructure. Nevertheless, to meet the reproducibility aspect researchers had to rely on additional services which required the duplication of entire datasets to guarantee the availability of an immutable version of the data. This approach, if complemented with software, computational environment, and explanatory metadata, can guarantee the research reproducibility but add not negligible costs, limitations and burden to the researcher: additional cost mainly relates to the need of disposing of disk space on a managed FAIR repository which guarantees long term storage and the maintenance of the storing infrastructure including backup and service availablity; limitations are related to the possibility of archiving large datasets on managed long term repositories; burden relates to the extra effort of researcher in setting up specific pre-processing requirements in reproducibility.

Additionally, the advances of sensing technologies and the large diffusion of geospatial data produce a stream of transactional operations (new data or high frequency updates) resulting in managing high-frequency changing datasets. In this case the previously mentioned solution to guarantee reproducibility may result in being particularly expensive and inefficient due to high duplication required: potentially a new snapshot for each experiment. Since Digital Object Identifier (DOI) is to be persistent by definition, in case of continuously varying datasets this cannot be used to reference an evolving dataset. Similarly to software development best practice, version numbers could be used as a unique identifier but it's certainly not meaningful for the users.

To facilitate the change analysis of data, SQL has introduced in its specifications methods to support the temporal query of versioned tables. This provides a standard interface for users to access data how they were in the repository on a specific time instant and how data changed in the repository within a specific time period. Unlike version numbers this approach, which involves the management of the system time at data storage level, is simpler and meaningful for the users. At the time of writing, OGC standards support *business-time* properties but do not offer any capability on managing temporal datasets with *system-time* features: for the above mentioned issues it constitutes a major barrier to reproducibility when secondary data from interoperable geospatial services are used in research.

Today data versioning solutions exist and vary in terms of technological approach and solutions depending on different data formats and deployment needs. Traditional data warehouses support temporal SQL only in a few cases and, in particular, PostgreSQL, which is the most diffused solution to manage spatial data thanks to its PostGIS extension, does not natively support data system-time. Big data solutions for generic file versioning, including geospatial file formats, exist and mostly rely on git or git-like solutions which permits the management of system-time data variation using git commits. While integrating useful transactional metadata, like committer name and commit message, this kind of solution are not transactional systems and may have issues of data consistency with

 doi:10.20944/preprints202312.2316.v1

concurrent write and read operations for multiple users. Most promising solution to this issue is the support of ACID transactions provided by LSTs solutions that use column-oriented formats to manage tabular data specifically designed for efficient storage and retrieval. Unfortunately, so far not many options to manage geospatial file formats exist.

From the presented state of the art we can conclude that it is clear that several applications that applies frequent changes to datasets exists and that to guarantee the reproducibility of the researches that are based on geospatial web services offering such a kind of datasets there is the need to support system-time capabilities within OGC standards. While a defined approach exists in SQL and LSTs to support system-time it is not yet currently adopted on commonly used storage solutions like those offered by the OSGeo's projects [79] and/or are easy to integrate in them without new software development. Together with the support of system-time it would be of great support for the availability of git-like metadata on user and motivation to permit: this would greatly support reproducibility and Open Science. In fact, it would not only allow us to retrieve temporal versions of the dataset but it would also permit us to perform data lineage analysis to fully understand the historical changes and better comprehend the dataset (including data provenance and ownership) with the effect of fostering the transparency and, ultimately, the credibility of science.

6. Conclusions and Future Directions

OGC Geospatial web services that are currently used in Spatial Data Infrastructures do not meet the reproducibility requirement set by Open Science since they do not guarantee the immutable access to a dataset in its status at a specific time of consumption. To support this capability, geospatial data management infrastructures should manage datasets versioning and Web services should expose these features to users. Since versions number may evolve extremely fast and are not meaningful to the user, the system time, which identifies the instant for which archived information had specific values, should be used, in conjunction with web service URL, as a unique identifier of the dataset. Currently, several data management tools that support system time functionalities exist but in most of the cases they do not support geospatial data type and functions or are not commonly integrated in Free and Open Source Software for geospatial web services. Due to the large and exponentially increasing diffusion and availability of open geo-datasets, the implementation of travel-time specifications to access continuous evolving states of datasets in OGC standards and technical solutions would greatly impact the Open Science movement by fostering research reproducibility and ultimately the science transparency and credibility. With respect to the current practice of extracting snapshots of used data and preserving a copy in permanent FAIR repositories, this solution will reduce the disk space, the redundancy of data and the cost of maintenance. Additionally, computational scripting to reproduce the research can be more easily created using standard requests.

From a technical perspective, new data management tools that support high throughput while guaranteeing data integrity in concurrent transactions are emerging as new solutions that meet in an easy and fast way the cloud requirement of data scalability and replicability. Such a kind of solutions already offer features of travel time but do not support geospatial data formats and geospatial indexing yet. This is certainly one of the solutions to be sought for the future due to their characteristics of offering analysis-ready datasets in so-called "cloud-native" solutions. Nevertheless, at the time of writing additional development and testing are still required to bring this solution into practice within the geospatial sector.

Author Contributions: The authors developed the different contributions: Massimiliano Cannata. Conceptualization; methodology; investigation; resources; visualization; writing – original draft preparation; writing—review and editing; supervision; project administration; funding acquisition. Collombin, Maxime. Investigation; writing – original draft preparation. Ertz, Olivier. Investigation; Writing—review and editing. Giuliani, Gregory. Conceptualization; Methodology; Writing—review and editing; writing – original draft preparation. Ingensand, Jens. Methodology; Conceptualization; writing—review and editing; writing – original draft preparation. Primerano, Claudio. Writing—review and editing; visualization; Strigaro, Daniele. Investigation; writing—review and editing.

Funding: This work was supported by swissuniversities in the frame of the *Programme Open Science: Measure A1* within the project "OSIReS".

Data Availability Statement: Data sharing not applicable

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ramachandran, R.; Bugbee, K.; Murphy, K. From Open Data to Open Science. *Earth Space Sci.* **2021**, *8*, e2020EA001562, doi:10.1029/2020EA001562.
2. Ai, Q.; Bhat, V.; Ryno, S.M.; Jarolimek, K.; Sornberger, P.; Smith, A.; Haley, M.M.; Anthony, J.E.; Risko, C. OCELOT: An Infrastructure for Data-Driven Research to Discover and Design Crystalline Organic Semiconductors. *J. Chem. Phys.* **2021**, *154*.
3. Kar, A.K.; Dwivedi, Y.K. Theory Building with Big Data-Driven Research – Moving Away from the "What" towards the "Why." *Int. J. Inf. Manag.* **2020**, *54*, 102205, doi:10.1016/j.ijinfomgt.2020.102205.
4. Qi, M.; Mak, H.-Y.; Shen, Z.-J.M. Data-Driven Research in Retail Operations—A Review. *Nav. Res. Logist. NRL* **2020**, *67*, 595–616, doi:10.1002/nav.21949.
5. Facts and Figures for Open Research Data Available online: https://research-and-innovation.ec.europa.eu/strategy/strategy-2020-2024/our-digital-future/open-science/open-science-monitor/facts-and-figures-open-research-data_en (accessed on 7 December 2023).
6. European Organization For Nuclear Research; OpenAIRE Zenodo: Research. Shared. **2013**, doi:10.25495/7GXK-RD71.
7. Felden, J.; Möller, L.; Schindler, U.; Huber, R.; Schumacher, S.; Koppe, R.; Diepenbroek, M.; Glöckner, F.O. PANGAEA - Data Publisher for Earth & Environmental Science. *Sci. Data* **2023**, *10*, 347, doi:10.1038/s41597-023-02269-x.
8. Żółtak, M.; Trognitz, M.; Ďurčo, M. ARCHE Suite: A Flexible Approach to Repository Metadata Management.; July 8 2022; pp. 190–199.
9. Re3data.Org UK Data Archive. **2012**, Over 8.000 data collections., doi:10.17616/R3088K.
10. Pampel, H.; Weisweiler, N.L.; Strecker, D.; Witt, M.; Vierkant, P.; Elger, K.; Bertelmann, R.; Buys, M.; Ferguson, L.M.; Kindling, M.; et al. Re3data – Indexing the Global Research Data Repository Landscape Since 2012. *Sci. Data* **2023**, *10*, 571, doi:10.1038/s41597-023-02462-y.
11. Chatenoux, B.; Richard, J.-P.; Small, D.; Roeoesli, C.; Wingate, V.; Poussin, C.; Rodila, D.; Peduzzi, P.; Steinmeier, C.; Ginzler, C.; et al. The Swiss Data Cube, Analysis Ready Data Archive Using Earth Observations of Switzerland. *Sci. Data* **2021**, *8*, 295, doi:10.1038/s41597-021-01076-6.
12. Farrell, E.; Minghini, M.; Kotsev, A.; Soler, G.J.; Tapsall, B.; Micheli, M.; Posada, S.M.; Signorelli, S.; Tartaro, A.; Bernal, C.J.; et al. European Data Spaces - Scientific Insights into Data Sharing and Utilisation at Scale Available online: https://publications.jrc.ec.europa.eu/repository/handle/JRC129900 (accessed on 1 September 2023).
13. Tandy, J.; van den Brink, L.; Barnaghi, P. Spatial Data on the Web Best Practices. *W3C Work. Group Note* **2017**.
14. Hobona, G.; Simmons, S.; Masó-Pau, J.; Jacovella-St-Louis, J. OGC API Standards for the Next Generation of Web Mapping. *Abstr. ICA* **2023**, *6*, 91.
15. Wang, C.; Yu, H.; Ma, K.-L. Importance-Driven Time-Varying Data Visualization. *IEEE Trans. Vis. Comput. Graph.* **2008**, *14*, 1547–1554, doi:10.1109/TVCG.2008.140.
16. Saracco, C.M.; Nicola, M.; Gandhi, L. A Matter of Time: Temporal Data Management in DB2 for z. *IBM Corp. N. Y.* **2010**, *7*.
17. Kedron, P.; Li, W.; Fotheringham, S.; Goodchild, M. Reproducibility and Replicability: Opportunities and Challenges for Geospatial Research. *Int. J. Geogr. Inf. Sci.* **2021**, *35*, 427–445, doi:10.1080/13658816.2020.1802032.
18. Konkol, M.; Kray, C. In-Depth Examination of Spatiotemporal Figures in Open Reproducible Research. *Cartogr. Geogr. Inf. Sci.* **2019**, *46*, 412–427, doi:10.1080/15230406.2018.1512421.
19. Giuliani, G.; Cazeaux, H.; Burgi, P.-Y.; Poussin, C.; Richard, J.-P.; Chatenoux, B. SwissEnvEO: A FAIR National Environmental Data Repository for Earth Observation Open Science. *Data Sci. J.* **2021**, *20*, 22, doi:10.5334/dsj-2021-022.
20. OGC OGC API - Features - Part 3: Filtering and the Common Query Language (CQL) Available online: https://portal.ogc.org/files/96288#simple-cql_temporal (accessed on 1 September 2023).
21. OGC Sensor Observation Service 2012.
22. OGC OGC API - Moving Features - Overview Available online: https://ogcapi.ogc.org/movingfeatures/overview.html (accessed on 1 September 2023).
23. Nüst, D.; Pebesma, E. Practical Reproducibility in Geography and Geosciences. *Ann. Am. Assoc. Geogr.* **2021**, *111*, 1300–1310, doi:10.1080/24694452.2020.1806028.

24. Cerutti, V.; Bellman, C.; Both, A.; Duckham, M.; Jenny, B.; Lemmens, R.L.G.; Ostermann, F.O. Improving the Reproducibility of Geospatial Scientific Workflows: The Use of Geosocial Media in Facilitating Disaster Response. *J. Spat. Sci.* **2021**, *66*, 383–400, doi:10.1080/14498596.2019.1654944.
25. KNIME Analytics Platform Available online: https://www.knime.com/knime-analytics-platform (accessed on 8 December 2023).
26. Yin, D.; Liu, Y.; Hu, H.; Terstriep, J.; Hong, X.; Padmanabhan, A.; Wang, S. CyberGIS-Jupyter for Reproducible and Scalable Geospatial Analytics. *Concurr. Comput. Pract. Exp.* **2019**, *31*, e5040, doi:10.1002/cpe.5040.
27. Sullivan, I.; DeHaven, A.; Mellor, D. Open and Reproducible Research on Open Science Framework. *Curr. Protoc. Essent. Lab. Tech.* **2019**, *18*, e32, doi:10.1002/cpet.32.
28. Kirchhoff, M.; Geihs, K. Semantic Description of OData Services. In Proceedings of the Proceedings of the Fifth Workshop on Semantic Web Information Management; Association for Computing Machinery: New York, NY, USA, Giugno 2013; pp. 1–8.
29. Blanc, N.; Cannata, M.; Collombin, M.; Ertz, O.; Giuliani, G.; Ingensand, J. OGC API STATE OF PLAY – A PRACTICAL TESTBED FOR THE NATIONAL SPATIAL DATA INFRASTRUCTURE IN SWITZERLAND. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2022**, *XLVIII-4-W1-2022*, 59–65, doi:10.5194/isprs-archives-XLVIII-4-W1-2022-59-2022.
30. Krishnamurthi, R.; Kumar, A.; Gopinathan, D.; Nayyar, A.; Qureshi, B. An Overview of IoT Sensor Data Processing, Fusion, and Analysis Techniques. *Sensors* **2020**, *20*, 6076, doi:10.3390/s20216076.
31. Strigaro, D.; Cannata, M.; Lepori, F.; Capelli, C.; Lami, A.; Manca, D.; Seno, S. Open and Cost-Effective Digital Ecosystem for Lake Water Quality Monitoring. *Sensors* **2022**, *22*, 6684, doi:10.3390/s22176684.
32. Zahumenskỳ, I. Guidelines on Quality Control Procedures for Data from Automatic Weather Stations. *World Meteorol. Organ. Switz.* **2004**, *955*, 2–6.
33. WMO, W.M. Climate Data Management System Specifications Available online: https://library.wmo.int/records/item/51447-climate-data-management-system-specifications (accessed on 6 October 2023).
34. Pozzoni, M.; Salvetti, A.; Cannata, M. Retrospective and Prospective of Hydro-Met Monitoring System in the Canton Ticino, Switzerland. *Hydrol. Sci. J.* **2020**, *0*, 1–15, doi:10.1080/02626667.2020.1760280.
35. Cannata, M.; Antonovic, M.; Strigaro, D.; Cardoso, M. Performance Testing of IstSOS under High Load Scenarios. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 467, doi:10.3390/ijgi8110467.
36. UFAM, U. federale dell'ambiente Le acque sotterranee utilizzate come acqua potabile Available online: https://www.bafu.admin.ch/bafu/it/home/themen/thema-wasser/wasser--fachinformationen/massnahmen-zum-schutz-der-gewaesser/grundwasserschutz/grundwasser-als-trinkwasser.html (accessed on 25 November 2023).
37. Haklay, M.; Weber, P. OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Comput.* **2008**, *7*, 12–18, doi:10.1109/MPRV.2008.80.
38. OpenStreetMap Statistics Available online: https://planet.openstreetmap.org/statistics/data_stats.html (accessed on 14 December 2023).
39. Mocnik, F.-B.; Mobasheri, A.; Zipf, A. Open Source Data Mining Infrastructure for Exploring and Analysing OpenStreetMap. *Open Geospatial Data Softw. Stand.* **2018**, *3*, doi:10.1186/s40965-018-0047-6.
40. Martini, A.; Kuper, P.V.; Breunig, M. DATABASE-SUPPORTED CHANGE ANALYSIS AND QUALITY EVALUATION OF OPENSTREETMAP DATA. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2019**, *IV-2-W5*, 535–541, doi:10.5194/isprs-annals-IV-2-W5-535-2019.
41. Merodio Gómez, P.; Pérez García, M.; García Seco, G.; Ramírez Santiago, A.; Tapia Johnson, C. The Americas' Spatial Data Infrastructure. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 432, doi:10.3390/ijgi8100432.
42. Sudmanns, M.; Augustin, H.; Killough, B.; Giuliani, G.; Tiede, D.; Leith, A.; Yuan, F.; Lewis, A. Think Global, Cube Local: An Earth Observation Data Cube's Contribution to the Digital Earth Vision. *Big Earth Data* **2023**, *7*, 831–859, doi:10.1080/20964471.2022.2099236.
43. Sudmanns, M.; Tiede, D.; Lang, S.; Bergstedt, H.; Trost, G.; Augustin, H.; Baraldi, A.; Blaschke, T. Big Earth Data: Disruptive Changes in Earth Observation Data Management and Analysis? *Int. J. Digit. Earth* **2020**, *13*, 832–850, doi:10.1080/17538947.2019.1585976.
44. Giuliani, G.; Chatenoux, B.; De Bono, A.; Rodila, D.; Richard, J.-P.; Allenbach, K.; Dao, H.; Peduzzi, P. Building an Earth Observations Data Cube: Lessons Learned from the Swiss Data Cube (SDC) on Generating Analysis Ready Data (ARD). *Big Earth Data* **2017**, *1*, 100–117, doi:10.1080/20964471.2017.1398903.
45. Lewis, A.; Oliver, S.; Lymburner, L.; Evans, B.; Wyborn, L.; Mueller, N.; Raevksi, G.; Hooke, J.; Woodcock, R.; Sixsmith, J.; et al. The Australian Geoscience Data Cube — Foundations and Lessons Learned. *Remote Sens. Environ.* **2017**, *202*, 276–292, doi:10.1016/j.rse.2017.03.015.
46. Dwyer, J.L.; Roy, D.P.; Sauer, B.; Jenkerson, C.B.; Zhang, H.K.; Lymburner, L. Analysis Ready Data: Enabling Analysis of the Landsat Archive. *Remote Sens.* **2018**, *10*, 1363, doi:10.3390/rs10091363.

47. Maso, J.; Zabala, A.; Serral, I.; Pons, X. A Portal Offering Standard Visualization and Analysis on Top of an Open Data Cube for Sub-National Regions: The Catalan Data Cube Example. *Data* **2019**, *4*, 96, doi:10.3390/data4030096.
48. Giuliani, G.; Masó, J.; Mazzetti, P.; Nativi, S.; Zabala, A. Paving the Way to Increased Interoperability of Earth Observations Data Cubes. *Data* **2019**, *4*, 113, doi:10.3390/data4030113.
49. Lankester, T.H. OpenGIS Web Map Services-Profile for EO Products, Version: 0.3. 3. **2009**.
50. Ferreira, K.R.; Queiroz, G.R.; Vinhas, L.; Marujo, R.F.B.; Simoes, R.E.O.; Picoli, M.C.A.; Camara, G.; Cartaxo, R.; Gomes, V.C.F.; Santos, L.A.; et al. Earth Observation Data Cubes for Brazil: Requirements, Methodology and Products. *Remote Sens.* **2020**, *12*, 4033, doi:10.3390/rs12244033.
51. van der Weide, T.; Papadopoulos, D.; Smirnov, O.; Zielinski, M.; van Kasteren, T. Versioning for End-to-End Machine Learning Pipelines. In Proceedings of the Proceedings of the 1st Workshop on Data Management for End-to-End Machine Learning; Association for Computing Machinery: New York, NY, USA, Maggio 2017; pp. 1–9.
52. Mathis, C. Data Lakes. *Datenbank-Spektrum* **2017**, *17*, 289–293, doi:10.1007/s13222-017-0272-7.
53. Haman, J.T.; Miller, C.G. *Introduction to Git*; Institute for Defense Analyses, 2022;
54. Kimball, R. Slowly Changing Dimensions. *Inf. Manage.* **2008**, *18*, 29.
55. Kulkarni, K.; Michels, J.-E. Temporal Features in SQL:2011. *ACM SIGMOD Rec.* **2012**, *41*, 34–43, doi:10.1145/2380776.2380786.
56. Jungwirth, P.A. TEMPORAL DATABASES: THEORY AND POSTGRES. Presented at the PGCon2019, 2019.
57. Soroush, E.; Balazinska, M. Time Travel in a Scientific Array Database. In Proceedings of the 2013 IEEE 29th International Conference on Data Engineering (ICDE); April 2013; pp. 98–109.
58. System-Versioned Tables Available online: https://mariadb.com/kb/en/system-versioned-tables/ (accessed on 20 November 2023).
59. IBM Documentation Available online: https://www.ibm.com/docs/en/db2-for-zos/13?topic=tables-temporal-data-versioning (accessed on 20 November 2023).
60. Deshpande, K. Oracle9i: Understanding Automatic Undo Management and Flashback Query. *Sel. J.* **2004**, *11*, 22–30.
61. Gregg, C. Familiar with Oracle Flashback Time Travel? If Not, Keep Reading…. Available online: https://blogs.oracle.com/dbstorage/post/familiar-with-oracle-flashback-time-travel-if-not-keep-reading (accessed on 19 November 2023).
62. rwestMSFT Create a System-Versioned Temporal Table - SQL Server Available online: https://learn.microsoft.com/en-us/sql/relational-databases/tables/creating-a-system-versioned-temporal-table?view=sql-server-ver16 (accessed on 20 November 2023).
63. Fearing, V. Periods and SYSTEM VERSIONING for PostgreSQL 2023.
64. Chiodi, P. Temporal Tables 2023.
65. Huang, S.; Xu, L.; Liu, J.; Elmore, A.; Parameswaran, A. OrpheusDB: Bolt-on Versioning for Relational Databases 2017.
66. Huang, S.; Xu, L.; Liu, J.; Elmore, A.J.; Parameswaran, A. Orpheus Db: Bolt-on Versioning for Relational Databases (Extended Version). *VLDB J.* **2020**, *29*, 509–538.
67. Sehn, T. When to Make a Dolt Commit | DoltHub Blog Available online: https://dolthub.com/blog/2022-09-28-when-to-dolt-commit/ (accessed on 26 September 2023).
68. Low, Y.; Arya, R.; Banerjee, A.; Huang, A.; Ronan, B.; Koepke, H.; Godlewski, J.; Nation, Z. Git Is for Data. In Proceedings of the Conference on Innovative Data Systems Research; 2023.
69. Kandpal, N.; Lester, B.; Muqeeth, M.; Mascarenhas, A.; Evans, M.; Baskaran, V.; Huang, T.; Liu, H.; Raffel, C. Git-Theta: A Git Extension for Collaborative Development of Machine Learning Models 2023.
70. Peuster, M.; Schneider, S.; Karl, H. The Softwarised Network Data Zoo 2019.
71. Park, K.-T.; Kim, H.-Y.; Kim, Y.-C.; Lee, S.-M.; Kim, Y.-K.; Kim, M.-J. Lake: Towards Highly Manageable Cluster Storage for Extremely Scalable Services. In Proceedings of the 2008 International Conference on Computational Sciences and Its Applications; June 2008; pp. 122–131.
72. Novella, J.A.; Emami Khoonsari, P.; Herman, S.; Whitenack, D.; Capuccini, M.; Burman, J.; Kultima, K.; Spjuth, O. Container-Based Bioinformatics with Pachyderm. *Bioinformatics* **2019**, *35*, 839–846, doi:10.1093/bioinformatics/bty699.
73. Vohra, D.; Vohra, D. Apache Parquet. *Pract. Hadoop Ecosyst. Defin. Guide Hadoop-Relat. Framew. Tools* **2016**, 325–335.
74. Liu, C.; Pavlenko, A.; Interlandi, M.; Haynes, B. A Deep Dive into Common Open Formats for Analytical DBMSs. *Proc. VLDB Endow.* **2023**, *16*, 3044–3056, doi:10.14778/3611479.3611507.
75. Camacho-Rodríguez, J.; Agrawal, A.; Gruenheid, A.; Gosalia, A.; Petculescu, C.; Aguilar-Saborit, J.; Floratou, A.; Curino, C.; Ramakrishnan, R. LST-Bench: Benchmarking Log-Structured Tables in the Cloud 2023.
76. Behm, A.; Palkar, S. Photon: A High-Performance Query Engine for the Lakehouse. *CIDR Www Cidrdb Org Httpcidrdb Orgcidr2022papersa100-Behm Pdf* **2022**.

 doi:10.20944/preprints202312.2316.v1

77. Add Geometry Type to Iceberg · Issue #2586 Available online: https://github.com/apache/iceberg/issues/2586 (accessed on 8 December 2023).
78. Hellman, F. Study and Comparsion of Data Lakehouse Systems. **2023**.
79. OSGeo Data Storage Projects Available online: https://www.osgeo.org/choose-a-project/information-technology/data/ (accessed on 15 December 2023).