Article

Processing Analytical Queries over Polystore System for a Large Astronomy Data Repository

Manoj Poudel *, Rashmi P. Sarode *, Yutaka Watanobe, Maxim Mozgovoy and Subhash Bhalla

Graduate Department of Computer and Information Systems, The University of Aizu, Aizu-Wakamatsu, Fukushima 965-8580, Japan; yutaka@u-aizu.ac.jp (Y.W.); mozgovoy@u-aizu.ac.jp (M.M.); c21bhala@u-aizu.ac.jp (S.B.)

* Correspondence: d8212102@u-aizu.ac.jp (M.P.); rashmipsarode@gmail.com (R.P.S.)

Abstract: There are extremely large heterogeneous databases in the astronomical data domain, which keep increasing in size. The data types vary from images of astronomical objects to unstructured texts, relations, and key-values. Many astronomical data repositories manage such kinds of data. The Zwicky Transient Facility (ZTF) is one such data repository with a large amount of data with different varieties. Handling different types of data in a single database may have performance and efficiency issues. In this study, we propose a web-based query system built around the Polystore database architecture, and attempt to provide a solution for the growing size of data in the astronomical domain. The proposed system will unify querying over multiple datasets directly, thereby eliminating the effort to translate complex queries and simplify the work for the users in the astronomical domain. In this proposal, we study the models of data integration, analyze them, and incorporate them into a system to manage linked open data provided by astronomical domain. The proposed system is scalable, and its model can be used for various other systems to efficiently manage heterogeneous data.

Keywords: ZTF data; PTF data; Polystore; link data; data integration

check for updates

Citation: Poudel, M.; Sarode, R.P.; Watanobe, Y.; Mozgovoy, M.; Bhalla, S. Processing Analytical Queries over Polystore System for a Large Astronomy Data Repository. *Appl. Sci.* **2022**, *12*, 2663. https://doi.org/10.3390/app12052663

Academic Editor: Sławomir Nowaczyk

Received: 7 January 2022
Accepted: 1 March 2022
Published: 4 March 2022

1. Introduction

On the web, there is a plethora of data sources. There are Smart cities, IoT and personal devices, human-curated datasets (e.g., Open Maps or Wikipedia), large-scale collaborative data-driven research, satellite data, and open government initiatives, which are all contributing to an enormous amount of data available for analysis. As a result of this diversity, related data can be found in a wide range of database engines.

The term "open data" refers to data that can be accessed, used, and shared by anybody. Open data can be used to help society, the economy, and the environment by governments, businesses, and individuals [1]. There are only two restrictions to open data: that it must be attributed and that it must be shared with others in a non-commercial manner. In order for the data to be accessible, it must be offered in its entirety and at a fair cost, preferably via the internet. Access to and modification of the data is also important. In order to allow for reuse and redistribution, the data must be made available under license terms that allow this, as well. Anyone can use, reuse, and redistribute; there should be no discrimination against fields of endeavor or individuals or groups [2].

The phrase "Linked Data" refers to a collection of best practices for publishing and connecting structured data on the Web. The data discovery technique is to access the data from integration tasks, such as schema matching, query reformulation, and mapping, from data lakes. It is about building the connections necessary for a person or machine to examine the data network [3]. When linked data are accessible, related data can be discovered. As with the hypertextual web, the data web is constructed using web documents. Unlike the web of data, where links are anchors for relationships in hypertext pages written in Hypertext Markup Language (HTML), data links are between arbitrary items represented by Resource Description Framework (RDF). Uniform Resource Identifier (URIs) can be used to identify any type of item or concept [4]. However, when it comes to HTML or RDF, the same requirements apply in order to help the web grow:

- Utilize URIs to identify objects;
- Utilize HTTP URIs to enable users to look up those names;
- When someone searches for a URI, give relevant information by utilizing industry standards (RDF, SPARQL);
- Include hyperlinks to additional URIs, so that users can continue to learn new things.

Nowadays, there is interest in integrating structured data with text, web pages, semi-structured data, as well as time series. Lots of data are available on the web in numerous kinds of formats. There is no DBMS (Database Management Systems) that performs well on all types of data. The "one size fits all" approach taken by FDBS (Federated Database Management System) is encountering challenges in delivering data management solutions for diverse data [5]. It is possible to have performance and efficiency issues if FDBS is used to manage several types of data. Distributed computing resources can be difficult to access, although web-based solutions are becoming increasingly available [6]. Multiple models and data stores are needed in today's database management trends. Data storage facilities work with a variety of data types which have their own local language.

Earlier data integration was accomplished through the use of a framework known as query discovery, the primary objective of which is to identify a query (or transformation) that translates data from one format to another. The objective is to identify the appropriate operators for joining, nesting, grouping, linking, and twisting data [7]. There is a strong emphasis on data science and analysis. Data science is frequently performed on vast repositories, sometimes referred to as data lakes, that contain a large number of distinct datasets [8]. The datasets may be sparsely or completely schema-less [9].

1.1. Polystore

The numerous data types and data sources available warrant an architecture that can leverage these data across multiple databases [10]. This has given rise to the concept of Polystores, which support transparent access across a variety of backends which have the disimilar data models. Polystores enable uniform querying across many data models and are necessary to handle data across numerous data models fast and efficiently. Polystores are a type of database management system that consists of a collection of interconnected heterogeneous database engines that communicate via an Application Programming Interface (API) [11]. Polystore systems, sometimes referred to as multistore systems, enable integrated access to a variety of heterogeneous cloud data storage, including NoSQL and RDMS. The taxonomy of a Polystore system was examined in a work [12] that categorized it as either loosely or tightly coupled. The mediator or wrapper concept is akin to the loosely coupled multistore architecture. When it comes to the data store, it has a common user interface and may be operated independently of the multistore architecture on a local level. Through an API, the wrapper interfaces with the data store to create queries, transform them, and execute them, and it delivers results to the operator engine. The tightly coupled multistore system enables local user engagement and task sharing across several systems, resulting in increased performance. Additionally, it enables the aggregation of data from several data sources. The hybrid system combines the benefits of a loosely coupled multistore with the advantages of a tightly coupled system. It optimizes querying many cloud-based data storage sources by using native subqueries and an operator order.

In order to manage capacity to access data/information between applications, polysources must link and be integrated to be processed. The changes in types of models and variety of data over the past few decades have given rise to extensive research on managing heterogeneous data. The data available in different scientific domains present challenges in integrating data with different models. Data integration is a process of integrating data from multiple sources [13]. It should provide a single view over all the sources. Generally, it features two types of architecture. Data Warehousing is a type of physical model in which data from multiple sources are copied and stored in a warehouse. The other type of architecture is the virtual architecture. The virtual architecture basically consists of Federated Database Systems (FDBS), a mediator, and the newly proposed Polystores.

1.2. Motivating Example of Linked Open Data in ZTF Repository

Astronomy is the science that investigates the physics, chemistry, and evolution of celestial objects and phenomena that originate outside the Earth's atmosphere, such as supernova explosions, gamma ray bursts, and cosmic microwave background radiation [14]. The four Vs of big data in the astronomy realm [15,16] are as follows: volume, variety, velocity, and value.

Volume refers to the quantity of data. Terabytes, petabytes, and even exabytes are used to describe data. As a result, big data present issues in terms of collection, cleansing, curation, integration, storage, processing, indexing, search, sharing, transferring, mining, analysis, and visualization. Conventional technologies are incapable of dealing with such massive amounts of data. Numerous ground- and space-based wide sky surveys are generating an avalanche of data in all fields of astronomy.

Variety is an indication of data complexity. Astronomical data are primarily comprised of images, spectra, time series, and simulation data. The majority of data are stored in catalogs or databases. The data from separate telescopes or projects are stored in their own formats, which challenges the analysis phase integration of data from disparate sources. Each data item contains a thousand or more features; this creates an issue of high dimensionality. Additionally, data come in a variety of formats: structured, semi-structured, unstructured, and mixed.

Velocity refers to the rate at which data are generated, transmitted, and analyzed. In terms of data volume, LSST will generate one SDSS per night for a period of ten years. Clearly, batch, stream, near-real-time, or real-time data analysis is required. LSST anticipates discovering 1000 new supernovae per night for ten years, implying that at least 10-100,000 notifications will be requested. How scientists will efficiently mine, correctly identify, and target supernova candidates, as well as conduct follow-up observations in ten years, are significant problems.

The term "value" refers to the data's great astronomical worth. It is exciting and inspirational to discover strange, rare, unexpected, and novel objects or occurrences in astronomy. Similarly, identifying a novel distribution trend or law is extremely valuable.

Numerous scientific data organizations have recently encountered the challenge of offering data management solutions for huge, heterogeneous datasets [17,18]. For instance, data from the Zwicky Transient Facility (ZTF) include relational data, imagery, light curves, and text. The ZTF data contain Epochal Science images and catalog files which further point to reference images and catalog files, so we can say that this falls under the category of linked open data. ZTF is a new time-domain survey that employs a wide-field survey camera to search for and analyze supernovae, variable stars, binaries, active galactic nuclei (AGN), and asteroids [19]. The ZTF is intended for the detection of near-Earth asteroids, unique and rapidly changing flux transients, and all types of variable sources in the Galactic plane [20]. The Intermediate Palomar Transient Factory (iPTF) project which was launched in the beginning of 2013 also features similar linkages between epoch science images and reference images, and is built on the heritage of the Palomar Transient Factory which is linked open data. This project was headed by Caltech (PTF) [21]. Through historical Palomar Transient Factory data and fast follow-up investigations of transient sources, iPTF enhanced tools for data reduction and source classification. In 2017, iPTF became the Zwicky Transient Factory, utilizing a rebuilt version of the same telescope that was used for iPTF. ZTF observations are made with a new camera that has a 47-square-degree field of view and is installed on the Samuel Oschin 48-inch (1.2 m) Schmidt telescope. The camera consists of 16 CCDs divided into four readout quadrants. As a result, each ZTF exposure generates 64 CCD quadrant images [19]. A CCD quadrant is the fundamental image-unit for pipeline processing and the source of all scientific data output. ZTF's large data will serve as a reference for the Large Synoptic Survey Telescope's next project (LSST) [22]. LSST will conduct surveys of location, flux and shape measurements, light curve analysis, and calibrated images. The PTF and ZTF projects' progress, as well as specifications regarding the products, are listed in Table 1.

Table 1. ZTF science data product.

Project Name	Duration	Data Download	No. of FITS File	Product
PTF (Level 0, Level 1)	2009–2012	0.1 TB per night	Around 3 million	Epochal images, photometric catalogs
iPTF (Level 2)	2013–2017	0.3 TB per night	Around 5 million	Deep reference, light curves
ZTF (Data release 1 to 8)	2017–2021	1.4 TB per night	Around 50 million	New reference, lightcurves, transient candidates, catalog
LSST (Data release 1 to 8)	2022–2024	3 TB per night	Around 500 million	Calibrated images, measure of position, flux and shapes, and light curves

All image data are available in the Flexible Image Transport System (FITS) format [23], which includes epochal (single-exposure) photos and photometric catalogs. The images were captured using 64 CCD (Charge-Coupled Device) cameras equipped with a variety of filters that affect the image quality [19]. The photometric catalogs provide information on images that comprise key-value pairs and header information. These key-value pairs can be converted to relations and stored in relational database management systems(RDBMS). Provided the data's size and the available resources, only a subset of the data is downloaded for indexing. Thus, the header files containing astronomical image information are downloaded. The header files are retrieved from the IRSA online service's Hierarchical File System (HFS) and replicated on the local server. Additionally, the catalogs include header files that include the metadata (HTML elements) necessary for establishing a connection to the IRSA web service and retrieving the images. PostgreSQL is used to store the key values and header information for the images obtained from IRSA/IPAC. The images are accessible via IRSA/cloud IPAC's service. Currently, the repository houses data for the years 2017–2021, comprising around 50 Terabytes.

As a result, astronomy has developed into a data-intensive science that continues to grow in size. Secure and scalable mass storage systems are in high demand for aggregate data in efficient processing [24]. Thus, data schemas may be modified often, and new data products can be added frequently. The data types include images of astronomical objects, unstructured texts, relations, and key-value pairs. To manage these data effectively, it will be necessary to categorize the sources of the data using extremely efficient machine learning algorithms. Significant increase across several identified sources will require the development of efficient and well-designed databases. The study is primarily motivated by the goal of developing a scalable multidata storage architecture for querying heterogeneous data which are available in polysources.

The enormous number and variety of data available in the astronomical realm present a significant management challenge. Currently available solutions in the astronomical area rely heavily on relational databases. Additionally, the majority of solutions require users to create sophisticated programs in order to gain significant insights. This project aims to propose a solution to astronomy's challenges with huge data and a dearth of appropriate query tools. The goals of this research are as follows:

1. Create an effective multidatabase architecture for managing heterogeneous astronomical data;

2. Provide a query language that federates data, transforms it, and migrates it effectively inside the underlying data repositories;
3. Incorporate a completely automated workflow-based query management system to manage heterogeneous data.

The rest of the paper is organized as follows: In Section 2, we discuss related work and existing work. In Section 3, we provide an overview of ZTF Data Processing and we describe how to Access ZTF data. In Section 4, we outline the Proposed System Overview of Open Data for Polystore Databases and we discuss the Workflow Web-Based Query Management System with Top-Down approach. In Section 5, we evaluate the existing work and features of Polystore systems with the Experimental setup and outline the discussion. Finally, in Section 6, we conclude with Summary and Conclusions.

2. Related Work

Unifying querying over different databases spanning various data models is a tedious task. The problem of accessing heterogeneous data from polysources has been researched in the context of multidatabase systems, Polystore systems, and data integration systems [6]. The existing datasets have a large amount of data obtained from various data sources. There can be multiple languages to access these different datasets. The bottom-up approach to making a Polystore considers language translation as the main task, as shown in Figure 1. The bottom-up approach is not preferable if a lot of data come from various sources and environments which support other languages and schemes. Thus, data and information in these kinds of systems create database connectivity issues or compatibility issues.

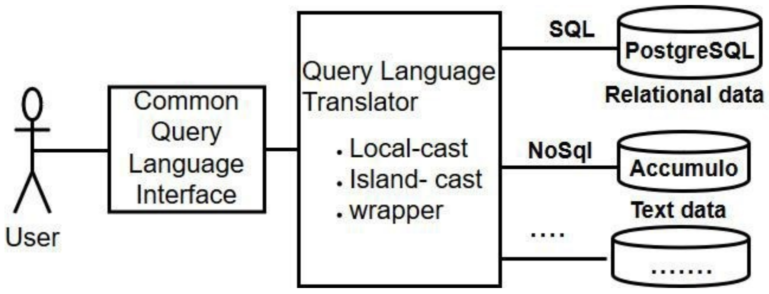

Figure 1. Bottom-up design of a Polystore system.

HYBRID is an analytical Polystore that is used to manage this type of data. HYBRID.POLY was created via a bottom-up methodology. HYBRID.POLY provides a unified platform for the storage, access, and analysis of several heterogeneous data types [25]. Numerous data models are supported by the in-memory storage engine. HYBRID.POLY's query interface allows queries written in a hybrid language. As a superset of SQL, the hybrid language can generate complex analytical queries on both nonrelational data (JSON, XML, and media files) and relational data. HYBRID optimizes the query and is capable of processing complex queries with a high number of nodes. The HYBRID.POLY query processing engine consists of a query parser, a query compiler, and a query optimizer. HYBRID.POLY utilizes a single unified data store that is capable of storing any form of data. This improves performance by lowering query processing and transmission costs.

The BigDAWG (Big Data Analytics Working Group) Architecture at MIT includes a query function for huge multiple data sets in the MIMIC II medical domain [26]. BigDAWG's architecture comprises four layers: database and storage engines, islands, middleware, application programming interfaces, and applications. It is aimed for a bottom-up strategy. BigDAWG's original release supported three free and open-source database engines: PostgreSQL (SQL), Apache Accumulo (NoSQL), and SciDB (NEW SQL). Additionally, it accommodates relational, array, and text islands. The client establishes communication with a Middleware or an API. The middleware intercepts a client query and forwards it to the relevant island or islands for execution. The Shim interprets queries received from each island and routes them to the appropriate database. Casts are used to migrate data across several databases [27].

The authors propose CloudMdsQL [28], a query language and query engine for cloud data. It is the golden mean between the two primary groups of query languages and combines the expressiveness of functional languages with the manipulation of declarative relational languages when it comes to integrating disparate data management systems. It also allows embedding invocations in native query interfaces. As it uses a distributed query engine, nodes can share code and data and there is no centralized wrapper/mediator. Small intermediate data can be processed by one node and they preserve query language expressivity. CloudMdsQL relies on the relational data model for join, union, and other relational algebra operations. They have integrated three DBMS—Sparksee (a graph database with a Python API), Derby (a relational database with a JDBC driver), and MongoDB for validation (a document database with a Java API). They demonstrated that CloudMdsQL meets the five important requirements for a cloud multiple-database query language by performing representative queries and showing the query engine's processing steps. It also optimizes query execution by rewriting queries using bind joins and pushed down selects, calculating optimal join execution orders, and transporting intermediate data efficiently. The results show overall query execution speed.

Polystore systems, particularly analytical Polystore systems, are becoming increasingly popular in modern big data applications because of their ability to handle a wide range of data sources and analytical tasks. ADIL, a dataflow language, is presented in this work [29], together with an AWESOME Polystore system. Powerful language ADIL supports heterogeneous data models such as Corpus and Graph; an extensive collection of analytical functions, and clear and strict semantics. With its support for the in-memory query engines and analytic capabilities, AWESOME is an efficient tri-store middleware that is built on top of three heterogeneous DBMSs (Postgres, Solr, and Neo4j). It uses a cost model to efficiently execute workloads written in ADIL, and fully utilizes machine resources to improve scalability. The capabilities, efficiency, and scalability of the AWESOME Polystore are demonstrated through a series of end-to-end drill-down experiments.

To analyze data, data scientists must sift through data lakes and enormous corporate databases, which may not be worthwhile tasks. In current data discovery solutions, users must manually seek relevant tables using multiple queries, which require understanding of the organization's schema and in some data lakes, this schema may not exist. The authors propose DICE (Data Discovery by Example) [30], a human-in-the-loop system that accepts user input and provides data that correspond to the user's intent. It is interconnected via numerous relational databases, and all tables are offline-indexed. DICE's central concept is to generate a SQL query that encapsulates the user intent, as stated through examples. DICE operates in a three-step manner. DICE begins by identifying a few possible queries by locating join paths between tables in the data lake. Then, by showing a few records to the user for validation, DICE eliminates unnecessary queries. DICE refines the search based on user feedback and continues the process until the user is satisfied with the results. The authors also explain how DICE will aid in data discovery by an interactive and example-based interaction.

2.1. Polystore System—Datawnt0 (Existing Work)

Datawnt0 is a top-down approach which uses a relational query language to query data from its headers database and subsequently access different resources. Datawnt0 supports a workflow-based query management system which uses a Polystore-based approach. It is a query language interface developed for skilled users. This system supports multi-object search to ease the work of the users belonging to the astronomical domain and also help these users by generating complicated programs [31]. This system uses a workflow method to make the queries understandable to the end user and also make the process of query execution faster. The multi-object search support also helps to perform a wider range of query execution. In Datawnt0, the user can relate multiple objects in the query as per the user's requirement. Every object selected during the query process generates a query component. Thus, multiple queries are added by a related objects operation and a complex query is formed, which can be easily executed by the Datawnt0 interface.

Palomar Transient factory (PTF) presents a rare example of a data store that is a shared resource for all astronomers [14]. The data are huge in size and variety. The data are routinely updated and new data (with a different new type) may be added. The primary objective of PTF project is to identify and classify astronomical objects, such as variable stars and supernovae, in real time. This project also aims to classify and query a catalog which contains other celestial objects for deeper analysis. Currently, there are three sets of data: Level-0, Level-1, and Level-2. Level-1 data include epochal (single-exposure) image and photometric catalogs. A single-exposure (sci) image provides CCD-quadrant (readout-amplifier)-based files. Level 2 covers all of the northern sky, including all the g- and r-band data, which are known as reference images. Reference images are co-adds of individual epochal images on a CCD-quadrant/filter basis that have been selected to satisfy the quality criteria. Reference image provides a static representation of the sky with good-quality of images for a given field, CCD quadrant, and filter [21,32]. Some drawbacks and limitations are addressed as follows:

- A limitation of Datawnt0 is that users can access only single-exposure (sci) images form PTF data archive;
- Another main drawback of Datawnt0 is that recursive queries do not work in this system as there is a loss of data. If a user needs to search entire fields and galaxies or search by position (galactic coordinates), the cDatawnt0 interface does not support this option at this time.

The limitations introduced by Datawnt0 can be addressed with proposed systems by providing users with Query by Example (QBE)-style language and a Set-Theoretic query language (relational algebra).

3. ZTF Data Processing Overview

The Infrared Processing and Analysis Center (IPAC) has developed PTF and ZTF Science Data Systems (ZSDS) [33]. These include data-processing pipelines, data archives, long-term curation infrastructure, and data retrieval user services. The ZSDS was created to provide a processing and archiving system capable of producing outputs of scientific grade. The data are analyzed in real time to provide real-time changes to the sky's information [20]. It is maintained as a repository for research works on numerous astronomical fields. These data are processed using a variety of processes to provide a variety of outputs for use in other scientific endeavors. There are nine pipelines, i.e., Raw Data Ingestion pipeline, Image Splitting pipeline, Calibration derivation pipelines (Bias-image Generation, Flat-field Image Generation), instrumental and photometrical pipeline, reference image pipeline, real-time image subtraction and extraction pipeline, light curve pipeline, and ZMODE: ZTF Moving Object Discovery Engine, which run on different timescales [19]. All astronomy data must be made publicly accessible to astronomers worldwide, as required by the grant agencies. These archival data are freely accessible to the public via the browseable web directory of the Infrared Science Archive (IRSA/IPAC) [34]. Through their online system, ZTF Science Exposure Metadata, calibration metadata, raw metadata, and reference image metadata can be accessed.

3.1. IRSA Archive

IRSA/IPAC curates and distributes the images and catalogs. Presently, there are around 6.9 million single-exposure images, 135,000 co-added images, 106 billion source catalog files, and 2 billion light curves generated from single-exposure extractions with already released catalogs [35]. The Image data are available in the Flexible Image Transport System (FITS) format as linked open data, which includes science images, calibration images, as well as the metadata of the calibration images, raw images, and reference images, which reference the actual images [23,36]. The ZTF archive web directory structure is shown in Figure 2. The IRSA preserves the ZTF FITS file in the B-tree data structure defined in prior work in their web directory, as in Table 2 . The root node of the IRSA web directory is called the top node, and the index is stored in the leaf node. The leaf nodes store the ZTF FITS image data where all insertions and updates occur.

The ith record contains an absolute identity as x_i to identify the base record. The components with x_i are year, two-digit month, and two-digit day, fractional day, field, filtercode, CCDID, image type code, quadrant ID, as shown in Figure 2. There are approximately 106 billion records in the ZTF archive. Each record has up to 1 billion data unit fields, which are identified as $y_{i,j}(y_{i,1} \ldots\ldots y_{i,1billion})$ where (i, j) indicate the record identity (x_i) and position of the field $((y_{ij})$ in the individual record.

Table 2. ZTF FITS file index in web archive.

Header	Data Unit	
x_i, y_{ij}, Size and index for the data (1, … 50 million)	y_{ij}	
	Name, size of Data	Data type
	Night, field, etc.	FITS, Log …

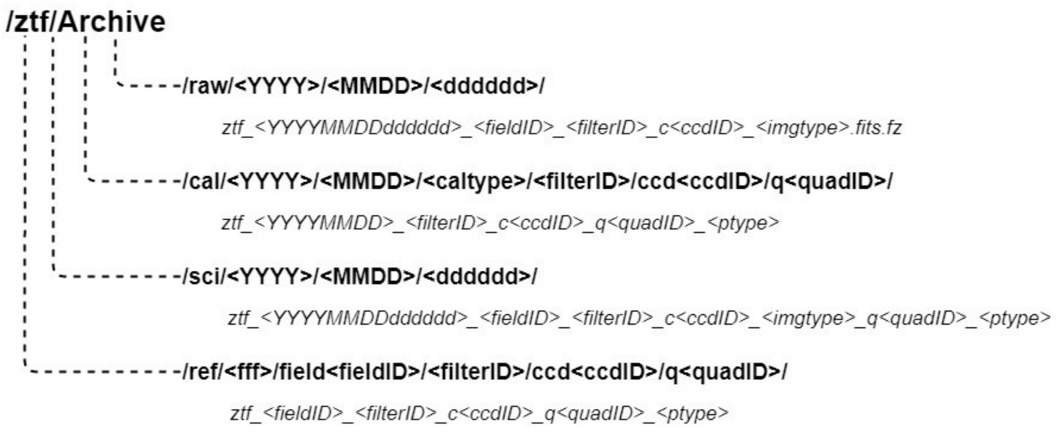
/ztf/Archive
/raw/<YYYY>/<MMDD>/<dddddd>/
ztf_<YYYYMMDDdddddd>_<fieldID>_<filterID>_c<ccdID>_<imgtype>.fits.fz
/cal/<YYYY>/<MMDD>/<caltype>/<filterID>/ccd<ccdID>/q<quadID>/
ztf_<YYYYMMDD>_<filterID>_c<ccdID>_q<quadID>_<ptype>
/sci/<YYYY>/<MMDD>/<dddddd>/
ztf_<YYYYMMDDdddddd>_<fieldID>_<filterID>_c<ccdID>_<imgtype>_q<quadID>_<ptype>
/ref/<fff>/field<fieldID>/<filterID>/ccd<ccdID>/q<quadID>/
ztf_<fieldID>_<filterID>_c<ccdID>_q<quadID>_<ptype>

Figure 2. ZTF Archive.

In the Figure 2,

- /raw = raw image data file;
- /cal = calibration product file;
- /sci = epochal science product file and difference images;
- /ref = reference image (co-adds) and catalog files;
- <fff> = first three (leftmost) digits of <fieldID>;
- <caltype> = calibration type, e.g., "bias", "hifreqflat";
- <imgtype> = single character label in raw camera image file;
- <ptype> = product type suffix string from pipeline;
- YYYY = year;
- MMDD = month and day (all UT-based);
- dddddd = fractional time of day of exposure (all UT-based);
- fieldID = 6-digit survey field ID if targeted science (on-sky) exposure, otherwise "000000" for calibrations;
- filterID = 2-letter filter code:zg, zr, zi for exposure acquired in g, R, or i respectively (for on-sky & flat-fields) bi, dk for bias and dark images, respectively (filter neutral);
- ccdID = 2-digit detector chip ID: 01, 02, … 16;
- quadID = 1-digit quadrant ID in ccd: 1, 2, 3, or 4 imgtype o: on-sky object observation or science exposure, b: bias calibration image, d: dark calibration image, f: dome/screen flatfield calibration image, c: focus image, g: guider image;
- ptype = fits, txt format image product type.

The API uses HTTP URLs to search and retrieve ZTF Image Metadata, Access Control, and ZTF Lightcurve Data. Users can construct the URL to obtain the image data product using the file path patterns described on the ZTF Metadata Page. Access Control is only for nonpublic data, which require a password. Access Control pages have general instructions for using the IRSA APIs with a password [37]. ZTF Lightcurve Data can be queried or retrieved using the IRSA ZTF-LC-API form of HTTP URLs via Specify ZTF objects by their identifiers (ID), by position (POS), by a collection of ZTF files (COLLECTION), by the format of output table (FORMAT) parameter [38]. The IRSA/IPAC directory has image URLs link indexed to support ZTF FITS files with API supports for visualization of images. This makes it easy to embed the archive directly in the user's software.

3.2. ZTF Released Products

IRSA has archived all raw metadata, processed data, and data archives and made them publicly accessible via data exploration tools. ZTF Image data products are made available to the public in a variety of formats and with a variety of features. All image data have been released in the Flexible Image Transport System (FITS) format, which includes CCD-based image metadata data files, CCD-quadrant-based image metadata data files, single-exposure science images, source catalogs, and Reference Images per CCD-quadrant. As indicated in Table 3, a CCD-quadrant is the fundamental image unit for pipeline processing, from which all scientific data outputs (DR1, DR2, DR3, DR4, DR5, DR6, DR7, and DR8) are created [39].

Table 3. ZTF science data product.

ZTF Public Data Released	Date of Releases	Data Acquired
Data Released 1 (DR1)	8 May 2019	ZTF-g and ZTF-r filters
Data Released 2 (DR2)	11 December 2019	ZTF-g , ZTF-r and ZTF-i filters
Data Released 3 (DR3)	24 June 2020	ZTF-g , ZTF-r and ZTF-i filters
Data Released 4 (DR4)	9 December 2020	ZTF-g , ZTF-r and ZTF-i filters
Data Released 5 (DR5)	31 March 2021	ZTF-g , ZTF-r and ZTF-i filters
Data Released 6 (DR6)	30 June 2021	ZTF-g , ZTF-r and ZTF-i filters
Data Released 7 (DR7)	8 September 2021	ZTF-g , ZTF-r and ZTF-i filters
Data Released 8 (DR8)	3 November 2021	ZTF-g , ZTF-r and ZTF-i filters

3.3. Access to ZTF Data

ZTF data can be retrieved, visualized, and analyzed from the file-based products, i.e., single-exposure science images or reference images, and their catalogs or other files using the IRSA ZTF Graphical User Interface (GUI) [40]. The IRSA provides two graphical user interfaces to access ZTF data, ZTF Images services, and catalog services. IRSA/IPAC support a download platform for these data through their browseable web directory.

Users can query the ZTF Images service using a graphical user interface to view and download ZTF images, search for astronomical objects by position, ZTF field ID, or solar system object/orbit. The IRSA image services implement a low-level search mechanism for astronomical images of metadata tables. IRSA image search services provide both single- and multi-object searches. The system's single object search function enables the user to look up astronomical bodies by their name or position. The query provides CCD-quadrant pictures that intersect these points, as well as metadata for further filtering. The result is shown in the web browser as a multicolumn table. The graphical user interface (GUI) enables previews of chosen images and interactive analysis. To perform a multi-object search, the user must manually build a sophisticated SQL query and upload it to the system. The system enables the user to load a file from a local disk or from one of the infrared scientific archive workstations.

Through three stages, users may retrieve, view, and evaluate catalog services using the GUI services. To begin, the user can search for an astronomical object using Single Object Search, Multi-Object Search, or All-Sky Search. The user may search for a single object using a single position and either a search radius or a box size centered on that place. After clicking on "Run Query", all objects and their associated light curves will be shown in a multicolumn table on a web browser based on user input. To obtain images from the multi-object search, the user must manually build a complicated query and upload it to the infrared science archive workplaces. The all-sky search retrieves counts from the full database table, whether it is in ASCII (IPAC-table) or HTML or XML format. The option perform an all-sky search does not return light curves.

The user can click on "Time Series Tool" on the results page that follows the query in step 1 to obtain the light curves of user input items. The light curves for all objects may be stored in ASCII (IPAC-table) format in this multi-column table [41]. The user can click on an individual object in the list of object IDs containing lightcurve data in step 2 and transmit its light curve to the Time Series Tool. These tools provide epoch-based science photos centered on an object, and include a period finder.

3.4. Retrieving the Catalog File from IRSA Remote Resource

The primary objective of the ZTF project is to identify and classify astronomical objects such as variable stars and supernovae in real time. This project also aims to organize and query a catalog that contains other celestial objects for more in-depth analysis. The output of astrometric calibration (ZTF CCD-quadrant images) is in FITS format. Each image in the FITS file includes a header data unit (HDU) and image data. HDU is observational metadata, which is the corresponding information linked to an image.

We have established a local repository for the purpose of the study. The IRSA remote server is used to obtain the table header information, imagery data, and key-value pairs for the tables. The download process was automated using a Python script that creates SQL dynamically, then optimized and saved in a PostgreSQL database on a local LINUX server with 16GB memory and 5TB disk space [42]. An SQL script assisted in optimizing and creating new tables in the local database with entity relations (key and value). The downloaded raw data are reorganized into a relational database structure (shown in Figure 3). These database tables are Nights, Exposures, Procimages, CCD, Filters, Fields, Host Fields, and Host Galaxy.

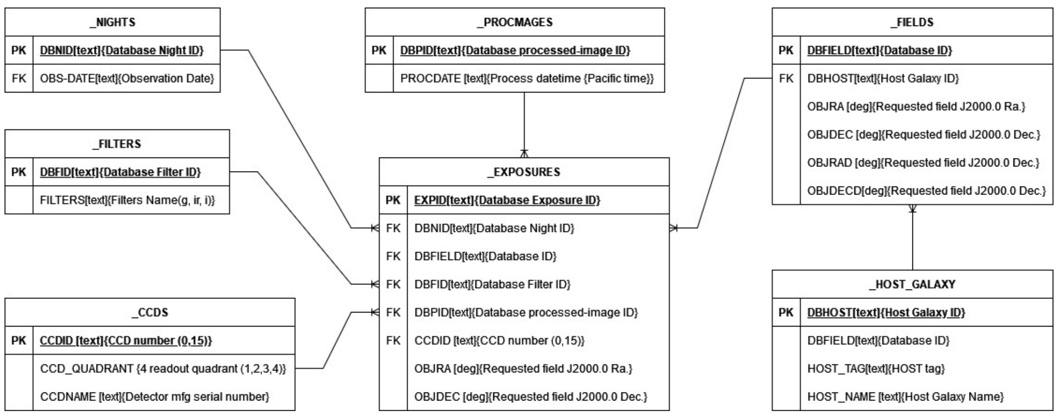
_NIGHTS
PK DBNID[text]{Database Night ID}
FK OBS-DATE[text]{Observation Date}
_PROCMAGES
PK DBPID[text]{Database processed-image ID}
PROCDATE [text]{Process datetime {Pacific time}}
_FIELDS
PK DBFIELD[text]{Database ID}
FK DBHOST[text]{Host Galaxy ID}
OBJRA [deg]{Requested field J2000.0 Ra.}
OBJDEC [deg]{Requested field J2000.0 Dec.}
OBJRAD [deg]{Requested field J2000.0 Dec.}
OBJDECD[deg]{Requested field J2000.0 Dec.}
_FILTERS
PK DBFID[text]{Database Filter ID}
FILTERS[text]{Filters Name(g, ir, i)}
_EXPOSURES
PK EXPID[text]{Database Exposure ID}
FK DBNID[text]{Database Night ID}
FK DBFIELD[text]{Database ID}
FK DBFID[text]{Database Filter ID}
FK DBPID[text]{Database processed-image ID}
FK CCDID [text]{CCD number (0,15)}
OBJRA [deg]{Requested field J2000.0 Ra.}
OBJDEC [deg]{Requested field J2000.0 Dec.}
_CCDS
PK CCDID [text]{CCD number (0,15)}
CCD_QUADRANT {4 readout quadrant (1,2,3,4)}
CCDNAME [text]{Detector mfg serial number}
_HOST_GALAXY
PK DBHOST[text]{Host Galaxy ID}
DBFIELD[text]{Database ID}
HOST_TAG[text]{HOST tag}
HOST_NAME [text]{Host Galaxy Name}

Figure 3. ER Model of the Database with Relations and Attributes.

- Nights: Nights contain the date or time of the images taken, with the unique index *nid* and alternative key—nightdate.
- Fields: Fields database table stores images according to the X and Y coordinate and assigned identification ID. In this table, *fieldid* is a unique index and *field* is an alternative key.

- Exposures: Exposure tables contain information from both night database table and fields database table. Exposure tables also contain detailed information on CCD, such as CCD ID, exposure time, image type, etc. The *expid* is a unique index and *obsdate* is an alternative key.
- Procimages: Procimages contains processed-images metadata, i.e., image file names. The unique index is processed-images number *pid*.
- Filters: Filters includes a record for each camera filter that was used to acquire the exposures. The unique index is *fid*.
- CCDS: CCD (16 charge-coupled devices) contains the camera details numbered CCDID 1, . . . 16. The unique index is *ccdid*.
- Host_galaxy: Host galaxy consists of the names of the galaxies.

4. Proposed System Overview and Use of Open Link Data Integration for Polystore Databases

Open Data Integration is proposed as a novel way of thinking about data integration across enormous, increasing data sources, with an emphasis on data exploration and discovery [43]. Integration is motivated by data analysis requirements, hence, data discovery is performed to aid in this analysis. Data analysis necessitates the discovery of data that precisely joins, unites, or aggregates with existing data—a paradigm termed "query-driven data discover". Some various databases and datasets are stored in multiple scientific data repositories.

Let us consider the following example to set up an integration application with multiple sources.

Two sources, DS_1 and DS_2, contain the data information. G_{12} is the match operator to create a schema mapping between source DS_1 and DS_2. M is the mediated schema by a schema merge operator to source DS_1 and DS_2 and the mapping G_{12}. The merge operator will create a schema that includes all the information in the source DS_1 and DS_2. The merge operator will also create mapping from source DS_1 and DS_2 to M, G_{1M}, G_{2M}, as seen in Figure 4.

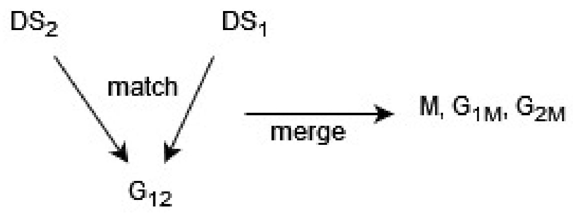

Figure 4. Common task in data integration with two sources.

Now, if we add another source, DS_3 a system and DS_3 is similar to DS_1, as seen in Figure 5.

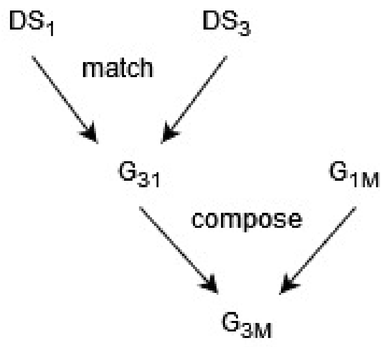

Figure 5. Composing mapping by adding new source.

From source DS_1 and DS_3, matching helps to create a mapping G_{31}. Additionally, by composing mappings G_{31} and G_{1M} helps to create a mapping G_{3M}.

To unify querying over different databases spanning a range of data models is a tedious task. Therefore, we propose a web-based Polystore system in this study to manage the complicated problem of heterogeneous data and to provide domain-specific search results across several storage systems. We save the header information, as well as the keys and values, on our local server using open-source PostgreSQL. All data and images pertaining to the ZTF are saved in the IRSA Cloud, which is built, managed, and hosted by IPAC. In the near future, this mapping can be implemented to connect different databases by employing top-down approaches.

4.1. Existing IPAC Sources

Astronomical domain specialists require query tools to gain access to a variety of data for a variety of astronomical entities. The most popular searches are for images and information on images. The user may have a variety of requests depending on the circumstance. Users may wish to query a single item or a collection of objects in order to obtain information. Additionally, the user may choose to include certain logical operators (and, or) between multiple objects in order to refine the query. In contrast, the other current solutions available for querying PTF data require the user to write complex programs in order to perform such complex queries. A comparison of the current system and the IRSA web system is presented in this research [44], where various queries from the time-domain astronomy are analyzed.

4.2. Proposed System

The proposed system is a web-based information system. HTML/CSS specifies the layout based on the user interface design. JavaScript and PHP both enable dynamic querying and multistage table querying. The proposed system's architecture is represented in Figure 6. The header information for each file is downloaded and saved on the Local Server (RDBMS), while the image file and all other associated data are kept on the IRSA cloud server. Thus, the proposed query system facilitates the conversion of keys to addresses and the migration of queries from the local server to the IRSA cloud server for image retrieval. Users can choose objects and enter values in the input box according to their search criteria in the proposed system.

The interface provides QBE and recursive query feature for time-based, position-based, and camera-specific ZTF data. Users can perform searches by entering details about a single object or a collection of objects, depending on the user's query. The multi-object search is enabled through the use of the relate and join operations in conjunction with numerous search choices.

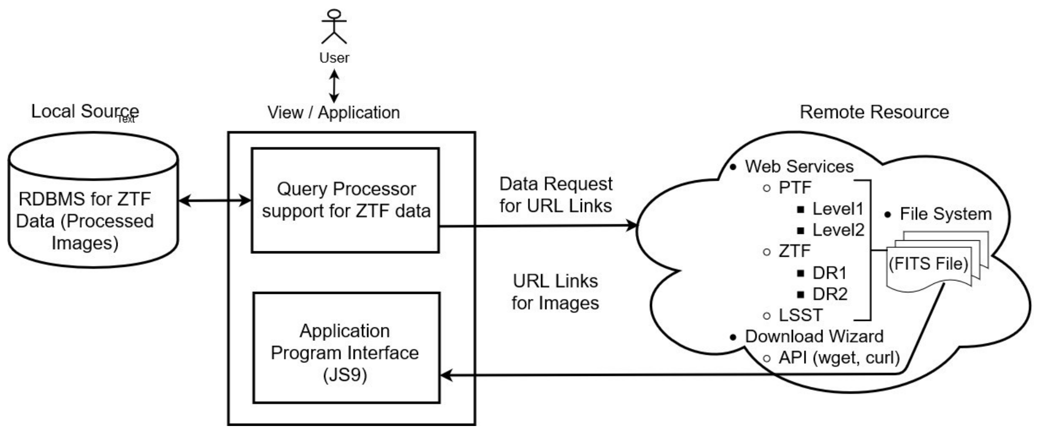

Figure 6. Proposed Web-Based Polystore System architecture.

The proposed system enables the formulation of queries using an interactive system in which users may select objects and input their associated IDs and predefined object names [23]. A query is created each time a user selects an object when querying. By combining these searches and linking many items, the basic query becomes more complicated. Multistage querying refers to the process of adding queries in many stages [45]. The server then stores the query, and fetches results in a table. Any desired row in the table can be selected so that the FITS image viewer JS9 [46] is connected and the images are visualized. The JS9 image viewer API defines a standard for communicating between local and distant data storage. The query language interface implements a set-theoretic query language for resolving relationships between objects of interest. It is built on the SQL query and is accessible to users who are unfamiliar with database programming.

4.3. Workflow Web-Based Query Management System with Top-Down Approach

As seen in Figure 7, the proposed web-based Information Requirements Elicitation (IRE) system employs a workflow mechanism for user convenience and includes an easily accessible query language. With the help of IRE, users can interact with the system to generate QBE to retrieve objects of interest. Mapping is used to transform the request and response between various database levels of architecture. Mapping is not suitable for small DBMS because it requires more time. The Conceptual/Internal Mapping lies between the conceptual level and the internal level. Its role is to define the correspondence between the records and fields of the conceptual level and files and data structures at the internal level. The External/Conceptual Mapping lies between the External level and the Conceptual level. Its role is to define the correspondence between a particular external and the conceptual view.

4.3.1. Schema Mapping

Schema mapping defines the conversion of data between the schema of an external source and the Integrate session schema [47]. This is depicted in Figure 8. Relational database tables and columns are mapped into classes and attributes in the session schema. The schema for the session is created from all data stores which have been accessed.

The top-down approach focuses on structuring and improving systems incrementally. Stepwise refinement enables the division of jobs into subtasks and, conversely, the refinement of data structures [48]. A good design prevents mistakes in multiple ways: it is resistant to frequent size changes, requires no translation, and reduces the cost of data integration. Since the ZTF data are open and well-documented, they serve as an excellent testbed for study on database techniques and performance. The research's top-down approach demonstrates a minor step toward the objective of offering superior data analysis and visualization capabilities for rapidly developing archives.

4.3.2. Proposal for Top-Down Polystore Systems Approach

The bottom-up approach to creating a Polystore considers language translation as the main task, as shown in Figure 1. The existing datasets contain a large amount of data obtained from various data sources. There can be multiple languages to access these different datasets. The bottom-up approach is not preferred if a lot of data come from various sources and environments which support other languages and schemes. Thus, data and information in these kinds of the system create database connectivity issues or compatibility issues.

The proposed system can support querying over multiple models uniformly and efficiently manage the information stored in various data models. For large and heterogeneous datasets such as ZTF, Polystores can provide top-down data management solutions. A framework for the top-down design process is shown in Figure 7. The activity begins with a requirement analysis that defines the system environment and data processing to create the database. For the data processing, we used two parallel activities: conceptual design and view design.

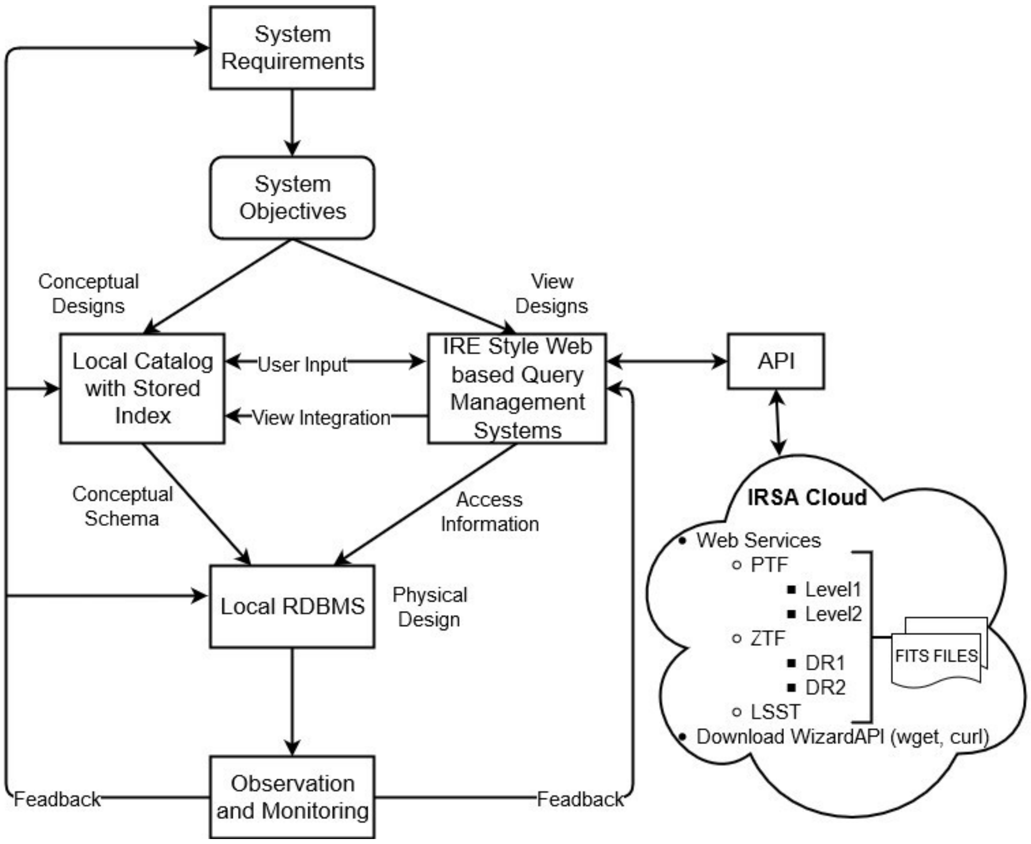

Figure 7. Web Polystore System top-down design approach.

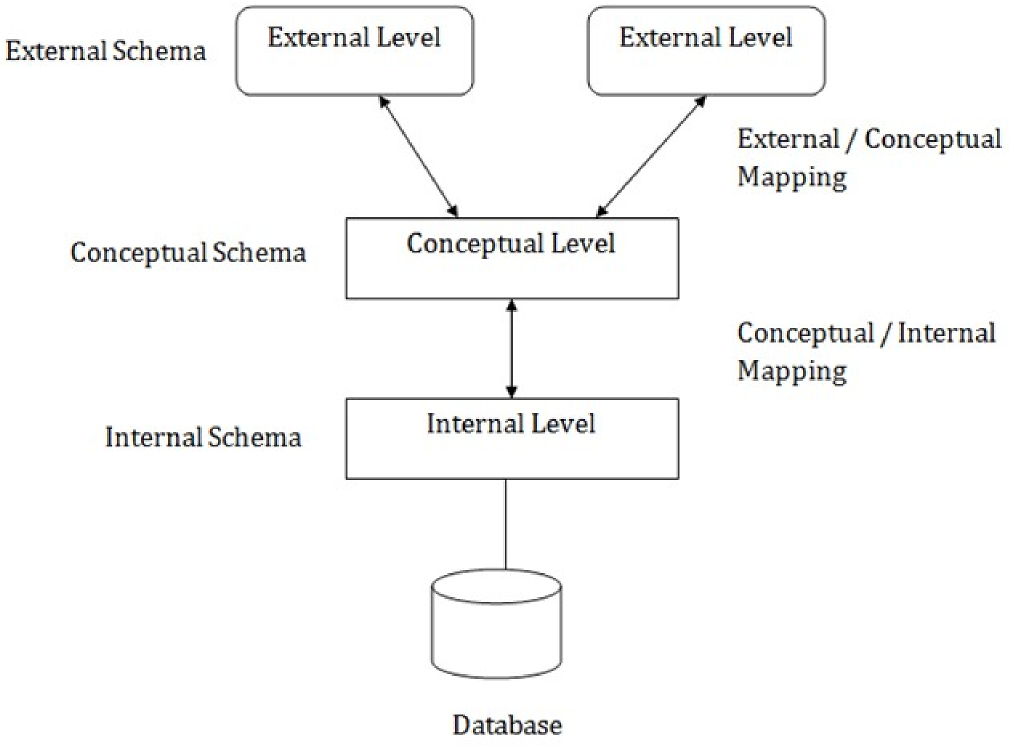

Figure 8. Schema mapping architecture.

The conceptual design stage is concerned with defining entity types and their connections. At the conceptual design stage, we select the ZTF catalog index file, the data in which should be stored in the database. These ZTF catalog index files were divided into separate entities, and the links between them were organized in the local RDBMS (Postgres). The database's fundamental entities are displayed as matrix tables in Table 4. The basic entities have different subclass, i.e., simple entity, complex entity, and N:N-link [49]. A simple entity is a collection of data about basic things recorded in a single database (for instance, HOST and FIELDS). A complex entity contains information on a complicated object that is logically maintained in several tables (for example, NIGHTS, FILTERS, FIELDS, CCD, and EXPOSURES). In a relational database, an N:N-link table is used to code N:N relationships (for example, HOST_FIELDS, EXPOSURE_CCD) where N:1 represents a many-to-one relationship and 1:N represents a one-to-many relationship. When expanding the conceptual model, the properties of each entity were determined and expanded as necessary by adding more indexes to the local catalogs.

Table 4. Matrix of the entities of the database.

Entities	Nights	Fields	Filters	Exposures	CCD	Host Galaxy	Procimages
Nights	-	-	-	1:N	-	-	-
Fields	-	-	-	1:N	-	1:N	-
Filters	-	-	-	1:N	-	-	-
Exposures	1:N	1:N	1:N	-	1:N	-	1:N
CCD	-	-	-	1:N	-	-	-
Host Galaxy	-	1:N	-	-	-	-	-
Procimages	-	-	-	1:N	-	-	-

The view design activity is concerned with the definition of end-user interfaces. The proposed system enables users to access ZTF data via a graphical user interface (GUI) that includes an image visualizer, via simple and complicated queries from the PostgreSQL database. The user has an option to select either science data product or reference data product by clicking on the sci or ref tab. The user interface of this system is created using HTML and CSS. JavaScript and PHP both offer dynamic and multistage table querying. Many systems can be extended from this architecture because more external views can be incorporated, and user workflow support can be added. The new additions or changes do not require query translation modifications. These only need to create dictionary entries at the conceptual level and support retrieval of these data items through an API at the data-server-side. The conceptual design might be understood as an integration of user perspectives. The activity of view integration is important since the conceptual design should accommodate both present and future applications [49]. The physical design links the conceptual schemas with real storage devices. At the physical design stage of this database, a data scheme physical model was constructed using the SQL relational database management system [50]. This database management system (DBMS) enables both basic and complicated query processing, with SQL created dynamically in response to user input with the graphical user interface (GUI). The generated query is then translated into predefined URL queries to the IPAC/IRSA web service in order to obtain the images.

4.3.3. Query Processor

The suggested web application has an integrated query processor for mapping queries, transforming data, and migrating it from the remote data store to the local database. SQL is used to communicate between the query processor and the web application and the underlying database management system. The user interaction with the system generates

SQL query statements that are handled by the Local RDBMS. The SQL query is then joined to the Processed Images (ProcImages) database, which contains the images' unique serial numbers. The query processor then compares the SQL query to the unique serial numbers of the images in the IRSA cloud, which are converted to URLs for the images. Finally, the SQL query is converted to server requests to obtain images from the multiple directory of IRSA data cloud, and matched images with the same unique serial number are presented in the web application's visualizer. Additionally, the query system enables users to link and combine objects in order to build a multi-object search. A query is created each time a user selects an object when querying. By combining those searches and linking many items, the basic query becomes more complicated and this process is depicted in Algorithm 1.

Algorithm 1: Query Workflow across the multiple data sources.

```
$i \leftarrow$ objects ;                              // Nights, Fields, CCDS, ...
$j \leftarrow$ attributes ;                           // nid, fid, ccdid, obj, ...
$k \leftarrow$ attributes properties ;  // nid=443,444.., fid=436, 836.., ccdid=0 16,
  ...
$n \leftarrow$ number of objects
for ( $i = 1; i <= n; i++$ ) do
    if ($k \in j_i$) then
        $R\,[\,] \leftarrow GeneratedImagesList$
        $satisfied \leftarrow TRUE$
    else
        $i++$ ;                                       // append with related object
    end
end
$R\,[\,] \leftarrow GeneratedImagesList$
$r = 1$
for each $r$ in $R\,[\,]$ do
    Select in $R\,[r]$ $\leftarrow$ SQL query is converted to server requests to obtain images
      from the IRSA cloud
end
```

Some sample queries that can be achieved are as follows:

- Query 1: Find the information of image from a Fields where user select example records from object list (e.g., fields, nights, exposures, procimages, ccd, etc.)
  SQL for Query 1:

  ```
  select distinct on (A."DBFIELD") A.* from "_FIELDS" A\\
  ```

  Image SQL for Query 1:

  ```
  select A.*,B.* from "_Exposures" A, "_PROCIMAGES" B,
  (select distinct on (A."DBFIELD") A.* from "_FIELDS" A)
  C where A."DBFIELD" = C."DBFIELD" and A."DBRID"= B."DBRID"
  order by B."DBPID" offset 0 limit 10
  ```

- Query 2: Find the information of image from a certain place field and exposures.
  SQL for Query 2:

  ```
  select distinct on (A."DBEXPID") A.* from "_exposures" A,
  (select distinct on (A."DBFIELD") A.* from "_FIELDS" A)
  B where A."DBFIELD"=B."DBFIELD"
  ```

  Image SQL for Query 2:

  ```
  select A.*,B.* from "_PROCIMAGES" B,
  (select distinct on (A."DBRID") A.* from "_exposures" A,
  ```

```
(select distinct on (A."DBFIELD") A.* from "_FIELDS" A)
B where A."DBFIELD"=B."DBFIELD" ) A where A."DBEXPID"= B."DBEXPID"
order by B."DBPID" offset 0 limit 10;
```

- Query 3: Find the information of image where field, night exposure is exactly the same in the tables.
  SQL for Query 3:

```
select distinct on (A."DBNID") A.* from "_NIGHTS" A,
(select distinct on (A."DBEXPID") A.* from "_EXPOSURES" A,
(select distinct on (A."DBFIELD") A.* from "_FIELDS" A)
B where A."DBFIELD"=B."DBFIELD" ) B where A."DBNID"=B."DBNID" ;
```

  Image SQL for Query 3:

```
select A.*,B.* from "_EXPOSURES" A, "_PROCIMAGES" B,
(select distinct on (A."DBNID") A.* from "_NIGHTS" A,
(select distinct on (A."DBEXPID") A.* from "_EXPOSURES" A,
(select distinct on (A."DBFIELD") A.* from "_FIELDS" A)
B where A."DBFIELD"=B."DBFIELD" ) B where A."DBNID"=B."DBNID" )
C where A."DBNID" = C."DBNID" and A."DBEXPID"= B."DBEXPID"
order by B."DBPID" offset 0 limit 10;
```

The proposed system can execute simple queries, such as Query 1, Query 2, and more complex queries, such as Query 3. Given the existing state of data access in ZTF, the current research can be focused on developing an alternative Top-Down Workflow Web-Based Query Management System for accessing ZTF data.

5. Evaluation and Discussion

In this Section, we evaluate the differences and similarities between the related query management systems, as discussed in Sections 2.1 and 4. For this purpose, we are considering Data Release 1 dataset for implementation. The query management system can be evaluated by various criteria such as data support, query support function, architecture flexibility, and users. The query management systems focus on providing a uniform/single query language from multiple heterogeneous data sources which have some differences and similarities.

5.1. Evaluation Based on Existing Work

Evaluation based on existing work is depicted in Table 5. We have compared IRSA ZTF Images GUI, Datawnt0 GUI with our Proposed Systems GUI. Datawnt0 does not have data support for reference images and catalog files, although it contains some features such as Multiobject search and it supports relate and join functions unlike IRSA. The proposed system has few parameters similar to IRSA, such as providing data support for reference images and catalog files and filtering option in result table, although it has some parameters such as QBE support and this system can be used by novice users (who do not have any knowledge of SQL), unlike IRSA and Datawnt0.

5.2. Evaluation Based on Features of Polystore System

The features of the proposed system have been compared to the existing systems which are BigDAWG and CloudMdsQL. We present this evaluation in Table 6. As opposed to existing systems that use islands/shims or mediators/wrappers to manage heterogeneity and multiple data stores, the proposed system uses APIs to process queries. The proposed system does not employ wrappers and instead leverages native API calls such as BigDAWG or CloudMdsQL to achieve autonomy. No data repositories are specified in the proposed system because it aims for ultimate transparency. Unlike existing systems that require data stores or information islands to be specified, the proposed method will automatically change into table representation and hide it. The proposed system lacks schema flexibility, although schema can be updated manually and the workflow is defined by standard and QBE SQL. Active query transformation with user-defined data migration is proposed as a way to build upon BigDAWG's data transformation and active data migration capabilities. These functionalities are not supported by CloudMdsQL.

Table 5. Evaluation based on Existing Work.

Evaluation Framework	IRSA ZTF Images GUI	Datawnt0 GUI (Past Work)	Proposed Systems GUI
Data support	• Reference images and catalog files • Epochal science images and catalog files	• Epochal science images and catalog files	• Reference images and catalog files • Epochal science images and catalog files
Query Support Function	• Single-Object Search • Multi-object Search, user upload, manually predefined table • Querying by Fields, CCDs, and Galaxies name • No Relate and Join Function • Filtering option present in the result table • No query by Example Function	• Single-Object Search • Multi-object Search • Query by Fields, CCDs, Nights • Support Relate and Join Function • Filtering option is not present in the result table • No Query by Example Function	• Single-Object Search • Multi-object Search • Query by Fields, CCDs, Nights, Galaxy name • Support Relate and Join Function • Filtering option present in the result table • Support Query by Example Function
Architecture Flexibility	• Local: Catalogs and images • Remote: Multiple directories with images and logs files	• Local: Catalogs with index • Remote: Single directory for images accessible	• Local: Catalogs with index • Remote: Multiple directory for images accessible
Users	• Expert SQL users	• Amateur SQL users	• Novice users

Table 6. Evaluation based on features of Polystore systems.

Evaluation Framework	BigDAWG	CloudMdsQL	Proposed System
Heterogeneity	• Multiple Data Stores with Multiple Query Interfaces • Query Processed through Islands and Shim Operators	• Multiple Data Stores with Single Query Interface • Query Processed through Mediators/Wrappers	• Multiple Data Stores with Single Query Interface • Query Processed through APIs
Autonomy	• Use of Shims, where catalog information updated automatically • Multi-object Search user upload manually predefined table • Native API calls	• Use of Wrappers where catalog updated automatically and manually • Native API calls	• Use of APIs where catalog updated automatically and manually • Native API calls
Transparency	• Specify information islands with SCOPE operators and hide transformation detail with CAST operators	• Specify data stores, data types and automatic transformation into table representation	• No need to specify data store, automatic transformation into table representation which is hidden
Flexibility	• No schema flexibility • Query interfaces of fixed islands, not readily extensible	• No schema flexibility • Subquerying and user defined MFR functions	• No schema flexibility (can be updated manually) • User defined workflow with standard and QBE SQL
Optimality	• Query Rewriting, data transformation and active data migration	• No active transformation and migration	• Active query transformation with user defined data migration

5.3. Experimental Setup

For the experiment, we selected 20 queries and evaluated the state of querying and analysis of large-scale astronomical data by comparing the current system with the IRSA web-based system. The evaluation confirms that the current query system can perform more queries and may be useful for the novice users who do not have query language knowledge. The most popular queries involve query by positions, query by observation date and time, query by host galaxies name, query by camera details, and query by example features.

- Query by positions: Uses galactic coordinates to specify the exact position to map the exact fields of galactic plane.
    1. Find all the objects in a certain galactic position.
- Query by observational date and time: Uses built-in calendar input function (OBJ-DATE) details and Night details which include the date (DD:MM:YYYY) and time (HH:MM:SS) per observed astronomical bodies.
    1. Find all the objects from a certain time period.
- Query by host galaxies: Uses target search for catalogs of nearby galaxies
    1. Find all the objects related to the specific galaxies.
- Query by camera details: Uses 16ccds cameras as per the different object filters used by ZTF, namely, zg, zr, zi for exposure acquired in g,R, I, respectively, and bi, dk for bias and dark images, respectively.
    1. Find all the object from camera filters;
    2. Find all the object from camera name.

5.4. Query Comparison Analysis

For the evaluation of the workflow-based query system, ZTF DR1 data were used and analyzed. This data set contains images, metadata with image header information, and relations. As stated in Section 3.3, the metadata and relations are saved in the local Postgres database. We downloaded data, made a schema and 20 queries were constructed for performance evaluation, as presented below.

1. Find all the images where Fields ID = 841;
2. Find all the images where Exposures ID = 44316126;
3. Find all images from fields where OBSJD = 2458197.6612616;
4. Find all the images where Night ID = 443;
5. Find all the images where Host galaxies where HOSTTAG = m81;
6. Find all images by observation date between 2018-04-01 and 2018-04-30;
7. Find all the images where Filters = 2;
8. Find all the images with R-band filters = zr;
9. Find all the images with CCD ID = 16;
10. Find all the images with Night ID = 443 and Fields ID = 809;
11. Find all the images with Night ID = 443 and CCD ID = 5;
12. Find all the images with Night ID = 443 and filtercode = zg;
13. Find all the images with Field ID = 841 and Exposure ID = 44316126;
14. Find all the images with Field ID = 809 and filtercode zg;
15. Find all the images where Exposures ID = 44316126 and Filters ID = 2;
16. Find all the images from date 2018-04-01 and 2018-04-30 and Field ID = 841 and CCD ID = 5 with R-band filters;
17. Find all the images with Night ID = 443 and Fields ID = 809 and CCD ID = 12 with g-band filter;
18. Find all the images from the Fields table or exposures tables;
19. Find all the Science Exposures images where Host galaxies name = m81;
20. Find all the References Images and Science Images where Host Galaxies name = ngc 13.

The proposed system and the source of ZTF data (IRSA web system) are used to validate the queries. The queries may involve the retrieval of data from a single or numerous objects. We have used all the possible queries by example, which is already predefined by the proposed system for those users who have a lack of query language knowledge (for novice users). The outputs of the queries were compared and depicted in Figure 9.

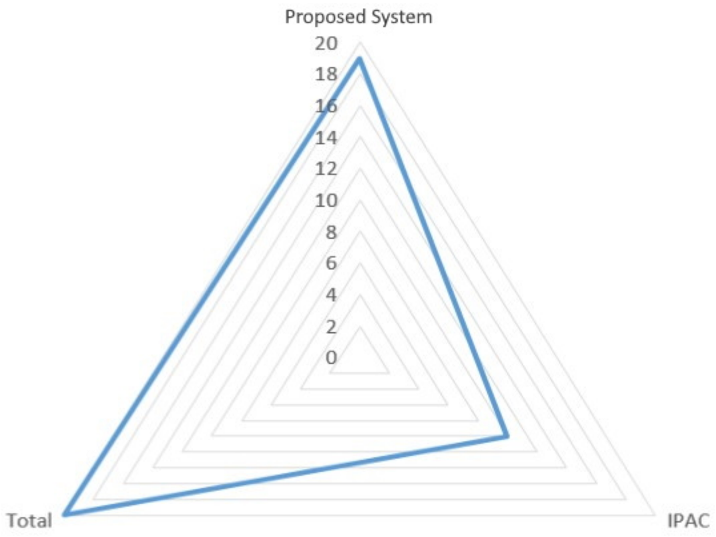

Figure 9. Query Comparison.

5.5. Discussion

Currently, businesses and database professionals encounter data challenges as data are enormous and a data management solution needs to be provided to manage these data [51]. Although Cloud storage is being used, it compromises security. Analysis of the existing scenario indicates a lack of proper query tools that support domain-specific language. A workflow-based query management system using a Polystore database approach is presented in the Proposed System. Data access to ZTF data using the Proposed System vis-a-vis the web based system of IRSA is compared. Experimental results show that querying is easier and more understandable using the workflow method. The proposed system also supports multi-object search where users from astronomical domain do not have to resort to complex programs. The range of queries supported is also wider due to the multi-object search support.

The rapidly growing archive of ZTF data warrants a top-down database design approach. The current research tries to incorporate these frequent changes in astronomical data (i.e., PTF, iPTF, ZTF, LLST) using a scalable Polystore architecture whose top-down approach facilitates the scalability issues through data discovery and reduction of data migration since the images are stored in the IRSA source. Finally in the absence of exact standard evaluation methods, the current architecture is contrasted with frameworks provided by past data integration models which are FDBS for heterogenity, autonomy and transparency as well as the ETL framework provided by Data Warehouse.

The volume and diversity of open data accessible in the astronomical sector presents a significant management problem when utilizing a top-down database design. The top-down design necessitates detailed data analysis in order to model conceptual and logical data. This data modeling process requires time and effort when working with large amounts of data. Another challenge arises when automating the extraction of vast amounts of open link data from the web.

The astronomical data archives are growing exponentially, which adds a variety of new data models. ZTF data also has graph data named Lightcurves used to analyze the light intensity of astronomical bodies or regions as a function of time. The graph data in ZTF present a plethora of avenues for future work. Future works include monitoring data integrity by comparing the query results of the query system with the results of IRSA web service. A variety of queries with different cases will be tested for precision, and the query system's recall will be studied. This will help in confirming the accuracy and integrity of the local query system.

Another future work that will be performed expeditiously is the integration of data from the Legacy Survey of Space and Time (LSST) into the local database. The LSST concept is to scan the sky deeply, widely, and rapidly using a single observing approach, resulting in data collection that meets the majority of the science goals simultaneously. The challenges associated with integrating a new database into an existing Polystore architecture can be surmounted.

The solution should additionally retain data integrity when compared to the original source of ZTF data. In future, we can also develop the precision and recall of the findings for the proposed solution that must adhere to the web system given by IRSA.

6. Summary and Conclusions

There are numerous types of astronomy data, and the amount of data available is enormous in repositories such as PTF, iPTF, ZTF, and LSST. ZTF has linked open data available in FITS format and we have used these data from ZTF repositories to create databases in order to unify them into a common query language. We have thus proposed a Workflow Web-based Polystore System Architecture based on a top-down approach rather than a bottom-up approach system that considers language translation as the main task. In the proposed system, the query processor processes the queries from different databases to unify them into a common relational query language. This system supports querying for multiple objects using the workflow method as a query is generated through user interaction in the GUI. The results are visualized in the form of tables and images. When an object is queried, the system generates two SQLs, one for the GUI and one for the FITS image viewer (visualizer) for linked open data in ZTF. The image SQL is used to connect to the remote image database which is transformed into server URL requests to fetch the images and display it in the visualizer through API. Additionally, we addressed current top-down and bottom-up approach-based systems, including Polystores. We assessed the Polystore system against existing works and also compared the proposed system's features to those of Polystore systems.

As the query system is built upon a Polystore architecture in the proposed system for integrated access to heterogeneous data, we have attempted to address the problem of scalability by integrating data and federating information via a top-down designed workflow web-based query management system.

Author Contributions: Supervision, Y.W., M.M. and S.B.; Writing—original draft, M.P.; Writing—review & editing, R.P.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. European Commission. What Is Open Data? 2007. Available online: https://data.europa.eu/elearning/en/module1/#/id/co-01 (accessed on 15 December 2021).
2. Handbook, O.D. What is Open Data? 2022. Available online: https://opendatahandbook.org/guide/en/what-is-open-data/ (accessed on 19 December 2021).
3. Nentwig, M.; Soru, T.; Ngonga Ngomo, A.C.; Rahm, E. Linklion: A link repository for the web of data. In Proceedings of the European Semantic Web Conference, Heraklion, Crete, Greece, 10–12 May 2014; pp. 439–443.
4. Berners-Lee, T. Linked Data. 2009. Available online: https://www.w3.org/DesignIssues/LinkedData.html (accessed on 1 January 2022).

5. Stonebraker, M. The Case for Polystores. 2015. Available online: http://wp.sigmod.org/?p=1629 (accessed on 11 December 2021).
6. Shrestha, S.; Bhalla, S. A Survey on the Evolution of Models of Data Integration. *Int. J. Knowl. Based Comput. Syst.* **2020**, *8*, 11–16.
7. Miller, R.J. Open data integration. *Proc. Vldb Endow.* **2018**, *11*, 2130–2139. [CrossRef]
8. Hai, R.; Quix, C.; Jarke, M. Data lake concept and systems: A survey. *arXiv* **2021**, arXiv:2106.09592.
9. Izquierdo, Y.T.; García, G.M.; Menendez, E.; Leme, L.A.P.; Neves, A.; Lemos, M.; Finamore, A.C.; Oliveira, C.; Casanova, M.A. Keyword search over schema-less RDF datasets by SPARQL query compilation. *Inf. Syst.* **2021**, *102*, 101814. [CrossRef]
10. Hota, L.; Dash, P.K. An Insight into Big Data and Its Pertinence. In *Handbook of Research for Big Data: Concepts and Techniques*; Apple Academic Press: New York, NY, USA, 2022; p. 137.
11. Duggan, J.; Elmore, A.J.; Stonebraker, M.; Balazinska, M.; Howe, B.; Kepner, J.; Madden, S.; Maier, D.; Mattson, T.; Zdonik, S. The bigdawg Polystore system. *ACM Sigmod Rec.* **2015**, *44*, 11–16. [CrossRef]
12. Valduriez, P. An Overview of Polystores. 2021. Available online: https://slideplayer.com/slide/13365730/ (accessed on 13 November 2021).
13. Doan, A.; Halevy, A.; Ives, Z. *Principles of Data Integration*; Elsevier: Amsterdam, The Netherlands, 2012.
14. Law, N.M.; Kulkarni, S.R.; Dekany, R.G.; Ofek, E.O.; Quimby, R.M.; Nugent, P.E.; Surace, J.; Grillmair, C.C.; Bloom, J.S.; Kasliwal, M.M.; et al. The Palomar Transient Factory: System overview, performance, and first results. *Publ. Astron. Soc. Pac.* **2009**, *121*, 1395. [CrossRef]
15. Bryant, A.; Raja, U. In the realm of Big Data. *First Monday* **2014**, *19*. [CrossRef]
16. Zhang, Y.; Zhao, Y. Astronomy in the big data era. *Data Sci. J.* **2015**, *14*, 11. [CrossRef]
17. Portela, F. Data science and knowledge discovery. *Future Internet* **2021**, *13*, 178. [CrossRef]
18. Shrestha, S.; Poudel, M.; Sarode, R.P.; Chu, W.; Bhalla, S. Open data integration model using a Polystore system for large scale scientific data archives in astronomy. *Int. J. Comput. Sci. Eng.* **2021**, *24*, 116–127. [CrossRef]
19. Bellm, E. The Zwicky transient facility. In *Third Hot-Wiring the Transient Universe Workshop*; IOP: Santa Fe, NM, USA, 2014; Volume 27.
20. Masci, F.J.; Laher, R.R.; Rusholme, B.; Shupe, D.L.; Groom, S.; Surace, J.; Jackson, E.; Monkewitz, S.; Beck, R.; Flynn, D.; et al. The zwicky transient facility: Data processing, products, and archive. *Publ. Astron. Soc. Pac.* **2018**, *131*, 018003. [CrossRef]
21. Kulkarni, S. The intermediate palomar transient factory (iptf) begins. *Astron. Telegr.* **2013**, *4807*, 1.
22. Bianco, F.B.; Ivezić, Ž.; Jones, R.L.; Graham, M.L.; Marshall, P.; Saha, A.; Strauss, M.A.; Yoachim, P.; Ribeiro, T.; Anguita, T.; et al. The Impact of Observing Strategy on Reliable Classification of Standard Candle Stars: Detection of Amplitude, Period, and Phase Modulation (Blazhko Effect) of RR Lyrae Stars with LSST. *arXiv* **2021**, arXiv:2108.01683.
23. Wells, D.C.; Greisen, E.W. FITS-a flexible image transport system. In *Image Processing in Astronomy*; Osservatorio Astronomico di Trieste: Trieste, Italy, 1979; p. 445.
24. Jiang, H.; Shen, F.; Chen, S.; Li, K.C.; Jeong, Y.S. A secure and scalable storage system for aggregate data in IoT. *Future Gener. Comput. Syst.* **2015**, *49*, 133–141. [CrossRef]
25. Elmore, A.J.; Duggan, J.; Stonebraker, M.; Balazinska, M.; Cetintemel, U.; Gadepally, V.; Heer, J.; Howe, B.; Kepner, J.; Kraska, T.; et al. A demonstration of the bigdawg Polystore system. *Proc. VLDB Endow.* **2015**, *8*, 1908. [CrossRef]
26. Clifford, G.D.; Scott, D.J.; Villarroel, M. User guide and documentation for the MIMIC II database. In *MIMIC-II Database Version*; Free Software Foundation: Cambridge, MA, USA, 2009; Volume 2.
27. Gadepally, V.; Chen, P.; Duggan, J.; Elmore, A.; Haynes, B.; Kepner, J.; Madden, S.; Mattson, T.; Stonebraker, M. The bigdawg Polystore system and architecture. In Proceedings of the 2016 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 13–15 September 2016; pp. 1–6.
28. Kolev, B.; Valduriez, P.; Bondiombouy, C.; Jiménez-Peris, R.; Pau, R.; Pereira, J. CloudMdsQL: Querying heterogeneous cloud data stores with a common language. *Distrib. Parallel Databases* **2016**, *34*, 463–503. [CrossRef]
29. Zheng, X.; Dasgupta, S.; Kumar, A.; Gupta, A. Processing Analytical Queries in the AWESOME Polystore [Information Systems Architectures]. *arXiv* **2021**, arXiv:2112.00833.
30. Rezig, E.K.; Bhandari, A.; Fariha, A.; Price, B.; Vanterpool, A.; Gadepally, V.; Stonebraker, M. DICE: Data discovery by example. *Proc. Vldb Endow.* **2021**, *14*, 2819–2822. [CrossRef]
31. Poudel, M.; Shrestha, S.; Sarode, R.P.; Chu, W.; Bhalla, S. Query Languages for Polystore Databases for Large Scientific Data Archives. In Proceedings of the 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 10–11 January 2019; pp. 185–190.
32. Patidar, R.G.; Shrestha, S.; Bhalla, S. Polystore Data Management Systems for Managing Scientific Data-sets in Big Data Archives. In Proceedings of the International Conference on Big Data Analytics, Warangal, India, 18–21 December 2018; pp. 217–227.
33. Duev, D.A.; Mahabal, A.; Masci, F.J.; Graham, M.J.; Rusholme, B.; Walters, R.; Karmarkar, I.; Frederick, S.; Kasliwal, M.M.; Rebbapragada, U.; et al. Real-bogus classification for the Zwicky Transient Facility using deep learning. *Mon. Not. R. Astron. Soc.* **2019**, *489*, 3582–3590. [CrossRef]
34. Caltech. NASA/IPAC Infrared Science Archive. 2021. Available online: https://irsa.ipac.caltech.edu/frontpage/ (accessed on 28 November 2021).
35. Caltech. Zwicky Transient Facility—Public Data Release 2. 2021. Available online: https://www.ztf.caltech.edu/news/public-data-release-2 (accessed on 25 November 2021).

36. Poudel, M.; Shrestha, S.; Yilang, W.; Wanming, C.; Bhalla, S. Polystore Database Systems for Managing Large Scientific Dataset Archives. In Proceedings of the 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 29–31 August 2018; pp. 1–6.
37. Bellm, E.C.; Kulkarni, S.R.; Graham, M.J.; Dekany, R.; Smith, R.M.; Riddle, R.; Masci, F.J.; Helou, G.; Prince, T.A.; Adams, S.M.; et al. The Zwicky Transient Facility: System overview, performance, and first results. *Publ. Astron. Soc. Pac.* **2018**, *131*, 018002. [CrossRef]
38. De, K.; Kasliwal, M.M.; Tzanidakis, A.; Fremling, U.C.; Adams, S.; Aloisi, R.; Andreoni, I.; Bagdasaryan, A.; Bellm, E.C.; Bildsten, L.; et al. The Zwicky Transient Facility census of the local universe. I. Systematic search for calcium-rich gap transients reveals three related spectroscopic subclasses. *Astrophys. J.* **2020**, *905*, 58. [CrossRef]
39. Laher, R.R.; Surace, J.; Grillmair, C.J.; Ofek, E.O.; Levitan, D.; Sesar, B.; van Eyken, J.C.; Law, N.M.; Helou, G.; Hamam, N.; et al. IPAC image processing and data archiving for the Palomar Transient Factory. *Publ. Astron. Soc. Pac.* **2014**, *126*, 674. [CrossRef]
40. Caltech. Zwicky Transient Facility—Mission Characteristics. 2021. Available online: https://www.ztf.caltech.edu/ (accessed on 23 November 2021).
41. Caltech. IPAC Table Format. 2021. Available online: https://irsa.ipac.caltech.edu/applications/DDGEN/Doc/ipac_tbl.html (accessed on 22 November 2021).
42. Wu, Y.; Chu, W. Query languages for domain specific information from ptf astronomical repository. In *International Workshop on Databases in Networked Information Systems*; Springer: Aizu-Wakamatsu, Japan, 2015; pp. 237–243.
43. OmniSci, I. Data Exploration—A Complete Introduction. 2021. Available online: https://www.omnisci.com/learn/data-exploration (accessed on 20 November 2021).
44. Shrestha, S.; Poudel, M.; Wu, Y.; Chu, W.; Bhalla, S.; Kupfer, T.; Kulkarni, S. PDSPTF: Polystore database system for scalability and access to PTF time-domain astronomy data archives. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*; Springer: Rio De Janeiro, Brazil, 2018; pp. 78–92.
45. Madaan, A.; Bhalla, S. Domain specific multistage query language for medical document repositories. *Proc. Vldb Endow.* **2013**, *6*, 1410–1415. [CrossRef]
46. JS9. JS9: Astronomical Image Display Everywheret. 2021. Available online: https://js9.si.edu/ (accessed on 24 November 2021).
47. Koleoso, T. Integrating with jOOQ. In *Beginning jOOQ*; Springer: New York, NY, USA, 2022; pp. 145–172.
48. Poudel, M.;Sarode, R.P.; Shrestha, S.; Wu, Y.; Chu, W.; Bhalla, S. Development of a Polystore Data Management System for an Evolving Big Scientific Data Archive. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*; Springer: Los Angeles, CA, USA, 2019; pp. 167–182.
49. Özsu, M.T.; Valduriez, P. *Principles of Distributed Database Systems*; Springer: New York, NY, USA; Dordrecht, The Netherlands; Heidelberg, Germany; London, UK, 1999; Volume 2.
50. Li, T.; Lockett, H.; Lawson, C. Using requirement-functional-logical-physical models to support early assembly process planning for complex aircraft systems integration. *J. Manuf. Syst.* **2020**, *54*, 242–257. [CrossRef]
51. Ponce, A.; Ponce Rodriguez, R.A. An analysis of the supply of open government data. *Future Internet* **2020**, *12*, 186. [CrossRef]