# Array DBMS: Past, Present, and (Near) Future

Ramon Antonio Rodriges Zalipynis
HSE University
Moscow, Russia
rodriges@gis.land

## ABSTRACT

Array DBMSs strive to be the best systems for managing, processing, and even visualizing big $N$-d arrays. The last decade blossomed with R&D in array DBMS, making it a young and fast-evolving area. We present the first comprehensive tutorial on array DBMS R&D. We start from past impactful results that are still relevant today, then we cover contemporary array DBMSs, array-oriented systems, and state-of-the-art research in array management, flavored with numerous promising R&D opportunities for future work. A great deal of our tutorial was not covered in any previous tutorial or survey article. Advanced array management research is just emerging and many R&D opportunities still "lie on the surface". Hence, nowadays we have the most favorable conditions to start contributing to this research area. This tutorial will jump-start such efforts.

**PVLDB Reference Format:**
Ramon Antonio Rodriges Zalipynis. Array DBMS: Past, Present, and (Near) Future. PVLDB, 14(12): 3186 - 3189, 2021.
doi:10.14778/3476311.3476404

## 1 OVERVIEW

The array DBMS R&D area is young and fast-growing, but is not yet widely known. It is also inherently inter-disciplinary: many core data types in geo-, bio-informatics, ecology, medicine, astronomy, to name a few, are naturally modeled by $N$-d arrays [3, 10, 40]. Array DBMSs are experiencing an R&D surge due to the rapid growth of big array data. For example, Maxar, a commercial company, alone acquires about 80 TB per day and accumulated over 100 PB of satellite data in AWS [37]. With this tutorial, we would like to increase awareness about this novel and exciting R&D direction, as well as to attract new researchers and inspire future work.

First array DBMSs and add-ons, e.g. RasDaMan [6], PostGIS [46], Oracle Spatial [57], appeared long ago. However, only the last decade flourished with a significant number of groups carrying out R&D on array management: ChronosDB (2018) [48, 49], SciDB (2008) [15] (array DBMSs); TileDB (2016) [44], SAGA (2014) [68] (array stores); DataCube (2017) [33], EarthServer (2016) [5] (national initiatives); Google Earth Engine (2012) [24], GeoTrellis (2012) [23], Dask (2018) [17] (array engines), and others [25, 48, 65].

Advanced array management approaches are just emerging. Novel indexing techniques accelerate array joins [73] and function evaluation [51]. Only recently, top-k queries [13], similarity array joins [77], and view maintenance [79] were first introduced. Work has just commenced on file-based sparse arrays [35] and caching [78]. Compression potentials are being explored [50] along with new formats for querying compressed arrays directly [30]. Array DBMSs begin to support interactive visualization [4, 49] and machine learning [43, 53, 65]. No existing survey covers those.

**Scope and structure**. We start from architectures, applications, data models, and query languages of array DBMSs. We also include array-oriented systems for completeness. As advanced array approaches are just emerging, we have a unique opportunity to cover the array management research accumulated to date and presented at major venues in sufficient depth. In particular, we cover state-of-the-art research in array storage, workloads, query execution, array indexing, joining, tiling, in-situ processing, machine learning, visualization, and benchmarking.

We show that the array DBMS R&D area can be viewed as young by right: no commonly accepted standards have yet been established, architectures and implementations still to be improved and matured, and many R&D opportunities are attractive and unexplored. In the text, we mark those by **R&D** to stimulate future work and promote starting novel research directions.

**Target audience**. The tutorial will be interesting to a broad range of researchers and practitioners working or about to work with array data in any scientific or applied domain. Although the array DBMS area is very young, we expect a large audience interested to attend this tutorial. Young researchers may consider this area for future work. More experienced attendees may discover many associations between array and relational DBMS research and find it tempting to apply their expertise in this area.

No special prerequisite knowledge is required for this tutorial. We will include the crucial information required for non-experts to commence research on the topic.

**Related tutorials**. As the array DBMS R&D area is rather young, array DBMS tutorials are quite sparse. The tutorial [65] briefly surveyed array-oriented systems (20 min.) and provided a vision for integrating machine learning and array DBMSs (70 min.). SciDB and RasDaMan tutorials were also given at XLDB [63] and BOSS [62] respectively. However, [62, 63, 65] do not cover research in array management to which our tutorial pays special attention. Existing survey articles are outdated [54] or totally omit research in array management [8]. In comparison, at least 70% of our tutorial was never summarized in any previous tutorial or survey article.

**Tutorial length**. We propose two versions. The 1.5 hours version will cover all material (the systems part will be short). The 3 hours version will contain an in-depth survey of array DBMSs, data models, operations and query languages (1.5 hours) while the algorithmic part will go to the second half of the tutorial (1.5 hours).

**Tutorial homepage**: http://vldb2021.gis.gg


Proceedings of the VLDB Endowment, Vol. 14, No. 12 ISSN 2150-8097.
doi:10.14778/3476311.3476404

## 2 TUTORIAL CONTENTS

### 2.1 Array DBMSs and Array-Oriented Systems

The story begins from TITAN [11], PARADISE [18], and RASDAMAN [6] which targeted Earth remote sensing data. For more than a decade, R&D in this area had been stalling. However, improved remote sensors & HPC simulation, cheap storage, and the resulting big array data avalanche has led to the renaissance of array DBMS R&D.

We start our tutorial from analyzing state-of-the-art array DBMSs and systems suitable for array processing cited in Section 1. Among other aspects, we will also answer the following questions. How do state-of-the-art array systems manage arrays? What are the main differences between these systems? What amount of effort and expertise is required for a user to start working with them? How to compose a query for an array DBMS and how clearly do queries express the intentions? Are array DBMSs interoperable with other software? What is the state-of-the-art array DBMS research? What are the promising R&D opportunities in this area?

### 2.2 Array Data Models and Query Languages

Array DBMS data models try to capture and generalize the diversity of big arrays: time-series (1-d arrays), $m \times n$ rasters (2-d arrays; $m, n \in S = \{x, y, time, height, model\ №, \dots\}$), $m \times n \times k$ spatio-temporal cubes ($k \in S$), and generally any $N$-d array.

The most widely used industry-standard array data models are Unidata CDM, GDAL Data Model and ISO 19123 [40]. They are mappable to each other and have evolved from decades of considerable practical experience. The most well-known research models and algebras for dense $N$-d, general-purpose arrays are AML [36], AQL [34], and RAM [64]: all are mappable to Array Algebra [7]. Recent work proposes data models optimized for geospatial [48] and bio-informatics domains [25]. Arrays are becoming ubiquitous [29].

Many array query languages were proposed [54], but only a handful of them survived. To date, operational array languages include AFL, AQL [15], rasQL [8], Command Line [48], GMQL [25], and array DBMS native UDF language [52]. In addition, an interesting ISO Array SQL [27] standard exists, but is not yet widely adopted. Array schema and query languages greatly differ between array DBMSs; there is no well-established practice yet [6, 15, 48].

**R&D** Array DBMS data models are at their early stages, largely focus on numeric arrays and omit other data types: polygons, tables, graphs, etc. It is challenging not just to jointly process diverse types, but to design a holistic, yet practical data model with an efficient implementation [1]. Many opportunities are open for designing new hybrid data models [32], algorithms [55], polystores [21], handling RDBMS tuples [29, 71] and graphs [20, 28] as arrays (tensors).

### 2.3 Array Storage

Array DBMSs serve diverse scientific communities that collected and processed array data for decades before array DBMSs appeared.

**Files**. Arrays natively come in diverse file formats, e.g. NetCDF-3, -4, HDF-4, -5, Grib-1, -2, GeoTIFF, BUFR, FITS [22, 42]. Formats are powerful storage containers that support chunking, compression, multidimensional arrays, and hierarchical namespace to name a few. Hence, a wide range of in-situ techniques leverage the formats' capabilities for optimized array processing, section 2.7.

**"Database approaches"** to array storage. TILEDB has a new on-disk format with support for fast updates [44], SCIDB targets ragged arrays [15], RASDAMAN keeps tiles in BLOBs [8]. In addition, HDFS array layouts exist [25]. Established file formats are evolving into "small databases", e.g. HDF-5 is equipped with indexing structures (e.g. B-Trees) and is being adapted for sparse arrays [35].

**Array compression** falls into 3 categories: (1) general-purpose, used not only for arrays; developed specifically for (2) scientific arrays or (3) array DBMSs. All 3 types are used by array DBMSs. For example, (1) ZLIB, BZLIB are built into SCIDB, CHRONOSDB leverages file-based compression techniques [50], bitmap compression [66] is extensively used in new in-db array layouts [72], (2) BitGrooming & Digit Rounding are used for NetCDF [75], available to array DBMSs, $k^2$-raster runs window queries directly on compressed arrays [30], (3) array DBMS compression is currently used for indexing [51, 73].

**R&D** Until recently, the R&D in array storage focused on persistent arrays. Emerging R&D reveals that specialized formats for interim arrays can vastly reduce I/O and save CPU time. For example, they can accelerate array joins [73] and function evaluation [51]. Moreover, there is a lot of work on NVM and RDBMSs [2], but NVM approaches proposed specifically for array DBMSs are lacking.

### 2.4 Array Operations (Workload Types)

Array workloads are rich and depend on the application domain, so array DBMSs provide a set of core operations [48] and allow users to code sophisticated array processing with UDFs [19, 52]. Researchers also optimize popular complex operations [13, 43, 76].

**Subsetting** (slicing or hyperslabbing) extracts an $(N - m)$-d subarray from an $N$-d array defined by hyperplanes. **Reshaping** reorders array axes. **Resampling** alters the resolution of array axes. At a glance, these operations may seem straightforward to execute. However, they have many complex subtypes (e.g. nearest-neighbor or Gaussian resampling methods) and executed in parallel. Hence, not all modern array DBMSs fully support all these operations [48].

**Aggregation** has many sub-types optimized separately: grid sliding, hierarchical, circular aggregations [68], co-addition [38].

**Map algebra** is an analysis language loosely based on the concepts presented in [61]. It is widely used in the industry [70]. It defines a rich set of array operations categorized into local, focal, zonal, and global types. These include algebraic computations, masking, IF-THEN-ELSE expressions, boolean and relational expressions, convolution, statistical aggregation, and other operations.

**Top-$k$ queries**. Overlap-allowing and disjoint top-$k$ subarray queries were first introduced in [13]. The output can be computed progressively and contains fixed-sized regions of an input $N$-d array sorted by a scoring function that satisfy some selection conditions.

**Histograms**. Efficient difference histogram construction methods were proposed in [76] that facilitate comparing observation datasets or spatial simulation datasets with different parameters.

**Array views** make it possible to reduce query storage footprints. Zhao et al. define view maintenance and propose a heuristic for effective view updates that come in batches at fixed time steps [79].

**Array caching**. Skewed array access workloads render existing array partitioning schemes inefficient and create load imbalance. The work [78] caches frequently accessed cells. It provides distributed algorithms for data placement, cache updates and eviction.

**UDFs**. Array DBMSs accept UDFs in Python/C++ which are black boxes that cannot be optimized [38]. UDFs amenable for window query optimizations are presented in [19]. The first native UDF language for array DBMSs is introduced in [48] and optimized via compiler techniques & strict formal definitions of array operations.

**R&D** For array DBMSs, novel applications are the major drivers for innovations [13, 38, 52]. For example, end-to-end physical world simulations can run directly inside an array DBMS equipped with new array management facilities [52]. Hence, many unexplored array DBMS applications are a fruitful ground for new challenges.

## 2.5 Array Join Techniques

Array joins substantially differ from relational table joins. Like RDMBS joins, the array join is always a hot research topic. Zhao et al. first formally defined array similarity joins and proposed load balancing algorithms for executing such joins [77]. Formally INNER and OUTER $K$-way array joins with respective algorithms were first introduced in [48]. Array joins also split into subtypes: equi-joins [56], dimensional- and value-based similarity joins [73, 77].

## 2.6 Array Tiling & Chunking Strategies

Large arrays, using a tiling strategy, are split into smaller, more manageable pieces to process them in small batches. Different tile/chunk shapes may yield orders of magnitude performance difference and are crucial performance parameters for array DBMSs [48].

Tiling/chunking can be regular, irregular, aligned, non-aligned, partially aligned, nested, with or without overlap [48, 54, 56]. Practitioners also worked-out diverse chunking strategies: equal scalar/dimension size, lefter product scalar size, balanced 1-d and $(N-1)$-d access to an $N$-d array, and others [41]. An array DBMS must be able to quickly alter tile/chunk shape (*re-tile* an array) to adapt to dynamic workloads [48]. Emerging re-tiling strategies address novel array DBMS applications and support novel array operations [52].

**R&D** Although tile shapes are important, it is usually not obvious a priori what shape is good in a given case. It is surprising, but shapes are mostly hand-tuned experimentally by array DBMS users nowadays. Novel heuristics are required for automatic selection of appropriate shapes (for disk & network I/O, load balance), especially in runtime for interim arrays whose shapes are not user-controlled.

## 2.7 In-situ Array Processing

Unlike the in-db approach, the in situ approach – one of the key array DBMS R&D trend – operates on data in their original file formats in a standard file system. As array data is rather big, the main advantage of in-situ techniques is the absence of a time-consuming import phase into an internal DBMS format. In addition, it is easier to share data in standard file formats with other systems.

Blanas et al. proposed in-memory techniques [9]. Over HDF5 files, SAGA runs aggregation queries [68] while ArrayUDF scales out user-defined sliding window functions [19]. DIRAQ reorganizes data for efficient range queries [31]. Su et al. proposed user-defined subsetting and aggregation over NetCDF files [59]. OLA-RAW performs parallel on-line aggregation of FITS files [12]. SciMate is optimized for several hyperslabbing patterns [67]. FastQuery and other bitmap indexes can be stored alongside the original data [14, 69]. ChronosDB performs all operations in-situ [48].

## 2.8 Array Indexing

We classify state-of-the-art array indexing types into 3 categories: (1) cell value selection (CS) and (2) hyperslabbing (HS): find cells in a given value range and spatial area respectively, (3) compute index (CI): accelerate computations over an array [9, 51, 72, 73].

Unlike RDBMS indexes, array DBMS indexes can be often used alone, without original data, to answer queries. As a great deal of array workload is I/O bound, the indexes usually aim to reduce I/O. Array DBMS indexes restructure data layout such that a query takes orders of magnitude less I/O compared to querying original data. In some sense, they resemble compressed data structures.

For example, consider a popular vegetation index $\text{SAVI} = (\text{NIR} - \text{RED})/(\text{NIR} + \text{RED} + L) \times (1 + L)$ [74]. NIR and RED are 2-d arrays with intensities of reflected solar radiation in the near-infrared and visible red spectra respectively, $L \in [0, 1]$ is a tunable parameter. Query examples – CS: select cells where $\text{SAVI} \in [0.7, 0.8]$, HS: select SAVI cells for Africa, CI: recompute SAVI for $L = L + 0.1$.

Equality (Eq), Range-Eq, and Interval-Eq bitmap indexes [9] are fast for CS queries, but they should be properly tuned or take excessive space and require extensive re-indexing on updates otherwise.

COMPASS efficiently executes both CS and HI queries [72, 73]. It partitions an array into regular chunks: integrated value indexes. A query (1) examines only chunks intersecting the query region, (2) reads the whole chunk or just buckets in the queried value range.

BitFun provides novel strategies to continuously re-index arrays during queries with similar mathematical functions [51], e.g. a CI query $\text{SAVI}(L + 0.1)$ may run 8× faster after computing $\text{SAVI}(L)$.

## 2.9 Array Visualization and Machine Learning

Visualization is crucial for data understanding. It is provided by array DBMSs to avoid costly data movements to external visualization systems. Battle et al. [4] used machine learning to predict future areas of user interest and render respective tiles beforehand in SciDB. ChronosDB features a novel distributed WMTS server [49]. RasDaMan delivers arrays over WCS and WCPS protocols [8].

Machine learning is just paving its way to array DBMSs [43, 53, 65]. Hence, this whole direction has endless **R&D** opportunities. Ordonez et al. [43] developed fast matrix multiplication algorithms that are orders of magnitude faster than Spark-based approaches. The work [53] addressed limitations of [43], including scalability. Linear algebra-based analytics using SciDB was evaluated in [60].

## 2.10 Array Database Benchmarks

Sequoia 2000 is one of the oldest DBMS benchmarks for arrays [58] extended later with additional queries [45] that are still relevant today. SS-DB is built on astronomical data [16]. Recent work evaluated systems on neuroscience and astronomy pipelines [38], while [48] thoroughly evaluates geospatial workloads. Single-node experiments on synthetic data were run in [8, 39]. Authors in [8, 26, 39] do not tune tile shape, a crucial performance parameter, section 2.6.

**R&D** Existing benchmarks hardly evaluate array DBMSs on skewed workloads, multi-tenancy scenarios, straggler nodes, heterogenous hardware (e.g., GPU, FPGA, NVM). Moreover, array DBMSs provide different sets of core operations, applications need new domain-specific operations. Hence, it is challenging to design a sufficiently generic benchmark: standardization efforts are needed.

## 3 PRESENTER DETAILS

Ramon Antonio is the author of ChronosDB array DBMS presented at VLDB 2018 [48] & SIGMOD 2019 [49], BitFun at VLDB 2020 [51], a novel R&D direction at SIGMOD 2021 [52], ChronosServer [47] and Climate Wikience wikience.org. He holds a Ph.D. in Ecological Safety (2013), M.S. (2008) and B.S. (2007) in Computer Science. He is currently an Associate Professor at the School of Software Engineering, Computer Science Faculty, HSE University. Ramon Antonio is the Best HSE Teacher for 5 years in a row (2017-21). Please, find more at hse.ru/en/staff/rodriges, gis.land, and gis.gg.

## REFERENCES


[1] Rana Alotaibi et al. 2019. Towards Scalable Hybrid Stores: Constraint-Based Rewriting to the Rescue. In *SIGMOD*. ACM, 1660–1677.
[2] Joy Arulraj and Andrew Pavlo. 2019. Non-volatile memory database management systems. *Synthesis Lectures on Data Management* 11, 1 (2019), 1–191.
[3] Venkatramani Balaji, Alistair Adcroft, and Zhi Liang. 2019. Gridspec: A standard for the description of grids used in Earth System models. In *arXiv*.
[4] Leilani Battle, Remco Chang, and Michael Stonebraker. 2016. Dynamic prefetching of data tiles for interactive visualization. In *SIGMOD*. 1363–1375.
[5] Peter Baumann et al. 2016. Big data analytics for Earth sciences: the EarthServer approach. *International Journal of Digital Earth* 9, 1 (2016), 3–29.
[6] Peter Baumann, Andreas Dehmel, Paula Furtado, et al. 1998. The multidimensional database system RasDaMan. In *SIGMOD*. 575–577.
[7] P. Baumann and S. Holsten. 2012. A comparative analysis of array models for databases. *Int. J. Database Theory Appl.* 5, 1 (2012), 89–120.
[8] Peter Baumann, Dimitar Misev, Vlad Merticariu, et al. 2021. Array databases: concepts, standards, implementations. *Journal of Big Data* 8, 1 (2021), 1–61.
[9] S. Blanas, Kesheng Wu, Surendra Byna, Bin Dong, and Arie Shoshani. 2014. Parallel data analysis directly on scientific file formats. In *SIGMOD*. 385–396.
[10] ArcGIS book. 2021. https://learn.arcgis.com/en/arcgis-imagery-book/
[11] Chialin Chang, Bongki Moon, Anurag Acharya, Carter Shock, et al. 1997. Titan: a high-performance remote-sensing database. In *ICDE*. 375–384.
[12] Yu Cheng et al. 2017. Bi-Level Online Aggregation on Raw Data. In *SSDBM*.
[13] Dalsu Choi, Chang-Sup Park, and Yon Dohn Chung. 2019. Progressive top-k subarray query processing in array databases. *PVLDB* 12, 9 (2019), 989–1001.
[14] Jerry Chou, Kesheng Wu, et al. 2011. FastQuery: A parallel indexing system for scientific data. In *IEEE International Conference on Cluster Computing*. 455–464.
[15] P. Cudre-Mauroux et al. 2009. A demonstration of SciDB: A science-oriented DBMS. *PVLDB* 2, 2 (2009), 1534–1537.
[16] Philippe Cudre-Mauroux, Hideaki Kimura, Kian-Tat Lim, Jennie Rogers, Samuel Madden, et al. 2010. SS-DB: A standard science DBMS benchmark. In *XLDB*.
[17] Dask 2021. https://dask.org/.
[18] David J DeWitt et al. 1994. Client-Server Paradise. In *VLDB*. 558–569.
[19] Bin Dong, Kesheng Wu, Surendra Byna, Jialin Liu, et al. 2017. ArrayUDF: User-Defined Scientific Data Analysis on Arrays. In *HPDC*. 53–64.
[20] F. Dörre et al. 2021. A GraphBLAS implementation in pure Java. In *GRADES*.
[21] Jennie Duggan, Aaron J Elmore, Michael Stonebraker, Magda Balazinska, et al. 2015. The BigDAWG Polystore System. *ACM SIGMOD Record* 44, 2 (2015), 11–16.
[22] HDF file format. 2021. https://www.hdfgroup.org/solutions/hdf5/.
[23] GeoTrellis 2021. https://geotrellis.io/.
[24] Noel Gorelick et al. 2017. Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment* 202 (2017), 18–27.
[25] Olha Horlova, Abdulrahman Kaitoua, and Stefano Ceri. 2020. Array-based Data Management for Genomics. In *ICDE*. 109–120.
[26] Fei Hu et al. 2018. Evaluating the open source data containers for handling big geospatial raster data. *ISPRS International Journal of Geo-Information* 7, 4 (2018).
[27] ISOSQLMDA 2019. SQL Part 15: Multi-Dimensional Arrays (SQL/MDA). https://www.iso.org/standard/67382.html.
[28] J. Kepner and J. Gilbert. 2011. *Graph algorithms in the language of linear algebra.*
[29] Dimitrios Koutsoukos et al. 2021. Tensors: An abstraction for general data processing. *PVLDB* 14, 10 (2021), 1797–1804.
[30] Susana Ladra et al. 2017. Scalable and queryable compressed storage structure for raster data. *Information Systems* 72 (2017), 179–204.
[31] Sriram Lakshminarasimhan et al. 2013. Scalable in situ scientific data encoding for analytical query processing. In *HPDC*. 1–12.
[32] Éric Leclercq et al. 2019. Polystore and Tensor Data Model for Logical Data Independence and Impedance Mismatch in Big Data Analytics. In *LNCS*. 51–90.
[33] Adam Lewis et al. 2017. The Australian Geoscience Data Cube—Foundations and lessons learned. *Remote Sensing of Environment* (2017), 276–292.
[34] Libkin et al. 1996. A query language for multidimensional arrays: design, implementation, and optimization techniques. *SIGMOD Record* 25, 2 (1996), 228–239.
[35] John Mainzer et al. 2019. Sparse Data Management in HDF5. In *XLOOP*. 20–25.
[36] Arunprasad P Marathe and Kenneth Salem. 2002. Query processing techniques for arrays. *VLDBJ* 11, 1 (2002), 68–91.
[37] Maxar 2017. 80 TB/day. https://youtu.be/mkKkSRIxU8M.
[38] Parmita Mehta et al. 2017. Comparative evaluation of big-data systems on scientific image analytics workloads. *PVLDB* 10, 11 (2017), 1226–1237.
[39] George Merticariu et al. 2015. Towards a general array database benchmark: Measuring storage access. In *Big Data Benchmarking*. 40–67.
[40] Stefano Nativi et al. 2008. Unidata's Common Data Model mapping to the ISO 19123 Data Model. *Earth Sci. Inform.* 1 (2008), 59–78.
[41] NCO. 2021. http://nco.sourceforge.net/.
[42] NetCDF. 2021. https://www.unidata.ucar.edu/software/netcdf/
[43] Carlos Ordonez et al. 2019. Scalable machine learning computing a data summarization matrix with a parallel array DBMS. *DPD* 37, 3 (2019), 329–350.
[44] S. Papadopoulos, Kushal Datta, Samuel Madden, and Timothy Mattson. 2016. The TileDB Array Data Storage Manager. *PVLDB* 10, 4 (2016), 349–360.
[45] Jignesh Patel et al. 1997. Building a scalable geo-spatial DBMS: technology, implementation, and evaluation. *ACM SIGMOD Record* 26, 2 (1997), 336–347.
[46] PostGIS 2021. http://postgis.net/.
[47] Ramon Antonio Rodriges Zalipynis. 2011. ChronosServer: real-time access to "native" multi-terabyte retrospective data warehouse by thousands of concurrent clients. *Inf., Cyb. and Comp. Eng.* 14, 188 (2011), 151–161.
[48] Ramon Antonio Rodriges Zalipynis. 2018. ChronosDB: Distributed, File Based, Geospatial Array DBMS. *PVLDB* 11, 10 (2018), 1247–1261.
[49] Ramon Antonio Rodriges Zalipynis. 2019. ChronosDB in Action: Manage, Process, and Visualize Big Geospatial Arrays in the Cloud. In *SIGMOD*. 1985–1988.
[50] Ramon Antonio Rodriges Zalipynis. 2019. Evaluating Array DBMS Compression Techniques for Big Environmental Datasets. In *IDAACS*, Vol. 2. 859–863.
[51] Ramon Antonio Rodriges Zalipynis. 2020. BitFun: Fast Answers to Queries with Tunable Functions in Geospatial Array DBMS. *PVLDB* 13, 12 (2020), 2909–2912.
[52] Ramon Antonio Rodriges Zalipynis. 2021. Convergence of Array DBMS and Cellular Automata: A Road Traffic Simulation Case. In *SIGMOD*. 2399–2403.
[53] Ramon Antonio Rodriges Zalipynis. 2021. Towards Machine Learning in Distributed Array DBMS: Networking Considerations *(LNCS)*, Vol. 12629. 284–304.
[54] Florin Rusu and Yu Cheng. 2013. A survey on array storage, query languages, and systems. *arXiv* (2013).
[55] S. Singla, A. Eldawy, R. Alghamdi, and M. Mokbel. 2019. Raptor: Large scale analysis of big raster and vector data. *PVLDB* 12, 12 (2019), 1950–1953.
[56] Emad Soroush, Magdalena Balazinska, and Daniel Wang. 2011. ArrayStore: a storage manager for complex parallel array processing. In *SIGMOD*. 253–264.
[57] Oracle Spatial. 2021. oracle.com/database/technologies/spatialandgraph.html.
[58] Michael Stonebraker, Jim Frew, Kenn Gardels, and Jeff Meredith. 1993. The Sequoia 2000 storage benchmark. *ACM SIGMOD Record* 22, 2 (1993), 2–11.
[59] Yu Su and Gagan Agrawal. 2012. Supporting user-defined subsetting and aggregation over parallel NetCDF datasets. In *CCGrid*. 212–219.
[60] Anthony Thomas and Arun Kumar. 2018. A comparative evaluation of systems for scalable linear algebra-based analytics. *PVLDB* 11, 13 (2018), 2168–2182.
[61] Dana C. Tomlin. 1990. *Geographic Information Systems and Cartographic Modeling.* New Jersey, US: Prentice-Hall.
[62] RasDaMan tutorial at BOSS. 2015. http://boss.dima.tu-berlin.de/2015/.
[63] SciDB tutorial at XLDB. 2013. http://rvernica.github.io/2016/07/tutorials.
[64] Alex van Ballegooij. 2004. RAM: A Multidimensional Array DBMS. In *EDBT*.
[65] Sebastian Villarroya and Peter Baumann. 2020. On the Integration of Machine Learning and Array Databases. In *ICDE*. 1786–1789.
[66] Jianguo Wang et al. 2017. An experimental study of bitmap compression vs. inverted list compression. In *SIGMOD*. 993–1008.
[67] Yi Wang, Wei Jiang, and Gagan Agrawal. 2012. SciMATE: A Novel MapReduce-Like Framework for Multiple Scientific Data Formats. In *CCGRID*. 443–450.
[68] Yi Wang, Arnab Nandi, and Gagan Agrawal. 2014. SAGA: Array Storage as a DB with Support for Structural Aggregations. In *SSDBM*. 1–12.
[69] Tzu-Hsuan Wei, Chun-Ming Chen, and Ayan Biswas. 2015. Efficient local histogram searching via bitmap indexing. In *EuroVis*, Vol. 34. 81–90.
[70] What is Map Algebra? – ArcGIS Help 2021. http://desktop.arcgis.com/en/arcmap/latest/extensions/spatial-analyst/map-algebra/what-is-map-algebra.htm.
[71] Hadley Wickham. 2011. The split-apply-combine strategy for data analysis. *Journal of statistical software* 40, 1 (2011), 1–29.
[72] Xing et al. 2018. COMPASS: compact array storage with value index. In *SSDBM*.
[73] Haoyuan Xing and Gagan Agrawal. 2020. Accelerating array joining with integrated value-index. In *SSDBM*. 145–156.
[74] Jinru Xue and Baofeng Su. 2017. Significant remote sensing vegetation indices: A review of developments and applications. *Journal of Sensors* (2017).
[75] C. Zender. 2016. Bit Grooming: statistically accurate precision-preserving quantization with compression. *Geoscientific Model Development* 9, 9 (2016), 3199–3211.
[76] Jing Zhao et al. 2018. Histogram Construction for Difference Analysis of Spatio-Temporal Data on Array DBMS. In *Australasian Database Conference*. 41–52.
[77] Weijie Zhao et al. 2016. Similarity join over array data. In *SIGMOD*. 2007–2022.
[78] Weijie Zhao et al. 2018. Distributed caching for processing raw arrays. In *SSDBM*.
[79] Weijie Zhao, Florin Rusu, Bin Dong, Kesheng Wu, and Peter Nugent. 2017. Incremental view maintenance over array data. In *SIGMOD*. 139–154.