27/05/2019

**Libraries and Framework**

**Rasdaman**

1. Raster data manager

Raster = a rectangular pattern of parallel scanning lines followed by the electron beam on a television screen or computer monitor

1. Open source
2. Open standards
3. Flexible and scalable
4. Cost saving
5. Fast
6. Rasdaman can process arrays residing in file system directories as well as in databases.
7. Array expressions return arrays or scalars.
8. Array queries return sets of arrays or scalars (like SQL).
9. Reading 3-D datacubes : trimming and slicing­.
10. How is an array actually returned?

Rasdaman returns it in the main memory format of the application ready for further processing.

1. Why is Rasdaman so fast? – Due to scalability
   1. Adaptive storage partitioning and distribution
   2. Distributed query processing – Queries are executed in parallel
2. Geo Raster Services
   1. Spatio-temporal data cubes
   2. Geo services implement OGC standards
   3. Allows space and time coordinates

**GeoTrellis**

1. It provides data types for working with rasters in the Scala language.

Coupling array queries with metadata query capabilities is of high practical importance; ISO SQL/MDA, with its integration of arrays into the rich existing framework of the SQL language, shows one possible way.

**rasdaman ("raster data manager")**

This array engine allows declarative querying of massive multi-dimensional arrays, including distributed array joins. Server-side processing relies on effective optimization, parallelization, and use of heterogeneous hardware for retrieval, extraction, aggregation, and fusion on distributed arrays.

 Arrays can be stored in the optimized rasdaman array store or in standard databases; further, rasdaman can operate directly on any pre-existing archive structure.

<https://www.rd-alliance.org/system/files/Array-Databases_final-report.pdf>

Array Databases Report

**Array Databases: Concepts, Standards, Implementations**

1. **Why Array Databases?**

In SQL, scientific and engineering environments could benefit only to a limited extent. The main reason is a fundamental lack in data structure support: While flat tables are suitable for accounting and product catalogues, science needs additional information categories, such as hierarchies, graphs, and arrays.

Array databases have worked out some key components of a powerful, flexible, scalable data management; these principles have proven successful over decades on sets (relational DBMSs), hierarchical data (e.g., XML databases), graph data (e.g., ontology and graph databases), and now array databases are offering their benefits as well.

Array databases provide flexible, scalable services on massive multi-dimensional arrays, consisting of storage management and processing functionality for multi-dimensional arrays which form a core data structure in science and engineering. They have been specifically designed to fill the gaps of the relational world when dealing with large binary datasets of structured information and have gained traction in the last years, in scientific communities as well as in industrial sectors like agriculture, mineral resource exploitation etc. 1-D sensor data, 2-D satellite and medical imagery, 3-D image timeseries, 4-D climate models are all at the core of virtually all science and engineering domains.

**Arrays can be stored in the optimized rasdaman array store or in standard databases; further, rasdaman can operate directly on any pre-existing archive structure.**

Concurrency and access control.

Avoiding inconsistencies due to parallel modifications of data is addressed by concurrency control with transaction support. Role based access control allows adjusting access for user groups individually.

Array database have been specifically designed to fill the gaps of the relational world when dealing with large binary datasets of structured information and have gained traction in the last years, in scientific communities as well as in industrial sectors like agriculture, mineral resource exploitation etc.

Implementations : **RASDAMAN, SciDB, SciQL**

Each of them allows querying the data based on the array’s properties and contents using declarative languages that usually allow for a large degree of flexibility in both query formulization and internal query optimization techniques.

1. **Array Database Concepts**
   1. **The Array Model**

Formally, a d-dimensional array is a function a: D → V with a domain consisting of the d-fold Euclidean cross product of closed integer intervals: D = {lo1, …, hi1} ∇ … ∇ {lod, …, hid} with loi≤hii for 1≤i≤d where V is some non-empty value set, called the array’s cell type.

Single elements in such an array are called cells.

* 1. **Querying Arrays**

Array Algebra relies on only three core operators: An array constructor, an aggregator, and an array sort operation.

The mdarray operator creates an array of a given extent and assigns values to each cell through some expression which may contain occurrences of the cell’s coordinate.

Suppose we want to obtain a subset of an array A. This subset is indicated through array coordinates, i.e., we extract a sub-array. For a d-dimensional array this subset can be defined through a d-dimensional interval given by the lower corner coordinate (lo1, ..., lod) and upper corner coordinate (hi1,...,hid), respectively. To create the subset array we write –

mdarray [ x( lo1:hi1, ..., lod:hid ) ]

elements a[x]

We can also reduce the dimension of the result by applying slicing in one or more coordinates. This means, instead of the loi:hii interval we provide only one coordinate, the slice position si. Notably, if we slice d times we obtain a single value (or, if you prefer, a 0-D array), written as:

mdarray [ x(s1, ..., sd) ]

elements a[x]

or in its shorthand

a[ s1, ..., sd ]

which resembles the common array cell access in programming languages.

Note : Trimming -- keeps the original dimensions

Slicing – reduces the number of dimensions

**Comparing rasdaman, SciDB, SciQL**

**Rasdaman**

Dimensions : n-D

array extensibility: all axes, lower and upper bound

null values : yes, null velue sets and intervals, can be assigned dynamically

relational tables : yes, via SQL/MDA std

horizontal spatial axes : yes

height/depth axis : yes

time axis : yes

Processing model : Declarative array QL

Formal semantics : Array algebra

Tightly integrated with any query language? : Yes, via SQL

**SciDB**

Dimensions : n-D

array extensibility: all axes, lower and upper bound

null values : yes (single null)

relational tables : no

horizontal spatial axes : no

height/depth axis : no

time axis : no

Processing model : Declarative array QL

**SciQL**

Dimensions : n-D

array extensibility: all axes, lower and upper bound

null values : yes, SQL-style (single null)

relational tables : yes

horizontal spatial axes : no

height/depth axis : no

time axis : no

Processing model : Declarative array QL

**SciQL**:

SciQL was a case study extending the column-store DBMS MonetDB with array-specific operators. As such, n-D arrays were sequentialized internally to column-store tables (i.e., there is no dedicated storage and processing engine).

**Tilaing:**

A core concept of array storage in rasdaman is partitioning or tiling. Arrays are split into sub-arrays called tiles for storage to achieve fast access. In the simplest case, tiles all are of the same size; however, more involved schemes are possible and may contribute substantially to better access performance once they mimic favourite access patterns (such as preferred subsetting along a time axis). Typically, systems implement regular tiling [17][20]; rasdaman, on the contrary, supports arbitrary tiling; actually, tiling is a tuning parameter for adjusting partitions to any given query workload, measured or anticipated, and minimize data access times.

After query analysis and optimization, the system fetches only the tiles required for answering the given query. Subsequent processing is highly parallelized.