Market Requirements Document

Feature Name: Advanced Array Handling

Version: 2 **Date Submitted:** 10/17/08 **Completed By:** Leon Guzenda
* Adds diagrams in Appendix A

Version: 1 **Date Submitted:** 10/14/08 **Completed By:** Leon Guzenda

Definitions (See Appendix A for diagrams)

- *Multi-dimensional array* - an n-dimensional matrix of elements with each row or column being of a fixed size. An element could be a primitive or a struct. For example, a matrix M with dimensions [2,3,4] would provide a cuboid of elements with sides 2, 3 and 4.
- *Array of arrays* – this is like a multi-dimensional array, but each array can have a varying number of elements. In Objectivity/DB terminology it would be represented as a VArray (or LVArray) of VArrays (or LVArrays).
- *Nested array* - a single or multi-dimensional array of elements that can themselves each be a single element (or struct), array, array of arrays, or nested array.

Usage

Algorithms that used meshes of points are widely used in scientific and engineering applications. Consider a flat mesh of points laid out on a rectangular grid and that the points have a latitude, longitude and temperature and are exactly one degree apart. An array with dimensions [-180:180,-90:90] and with each cell containing a temperature will represent the mesh and be easily accessed by latitude and longitude, e.g. Temperature[10,80].

Now suppose that there is a mesh for each day of the year. We can add another dimension and have an array with dimensions [365,-180:180,-90:90], or 366 in a leap year. So, a multi-dimensional array works fine for this particular set of data. An alternative would be to create a Year array of fixed dimension 365, 366 or with a variable dimension and with each element pointing to a Temperature array. This is an array of arrays.

In a real system it is likely that there are more readings in some cells than in others, so it is convenient to divide a cell into smaller cells, creating a two dimensional mesh within some elements. This is hard to declare in a static language, but easy to create in a dynamic language. For instance, it might be possible to divide a cell into a 50x50 mesh like this: Temperature[10,80] = new(Temperature,50,50) and then say Temperature[10,80][1,1]=todays_value etc.

We now have a nested array and we could further subdivide cells and cells within those cells to suit the granularity of the observations being recorded in the array.

This may look messy to a database practitioner, who would probably prefer to see collections of collections of simpler objects, but the fact is that scientists are used to working with array manipulation languages and they want to remove the impedance mismatch between their mathematical algorithms and the database.

Problems

1. Although Objectivity/DB supports objects with multiple varying length arrays (VArrays) it does not directly support arrays of arrays or nested arrays.
2. Objectivity/DB VArrays and LVArrays cannot currently scale beyond virtual memory constraints (though there is a separate MRD covering both chunked and streamed VArrays and LVArrays).

Description of the Requested Feature

1. The ability to declare multi-dimension arrays, arrays of arrays and nested arrays as regular objects or components of objects.
2. The ability to create, access, change and delete the new structures with semantics appropriate to the interface language (e.g. C++, Java and Matlab).
3. The ability to read and update elements within the new structures.
4. The ability to create and delete composite (array, array of array or nested array) elements.
5. The ability to extend or truncate arrays.
6. The ability to copy arrays and slices of arrays (whose elements may have dependents) into compatible structures. Note: See Caveat 2 below.
7. The ability to perform pointer arithmetic on individual arrays (when the interface language supports it).
8. Individual arrays must not be bound by the limitations of available virtual memory, i.e. it must be possible to access or stream groups of contiguous elements. Note: This requirement is also covered by a separate MRD.
9. The ability to control the caching characteristics for a structure, e.g. move no more than N elements into memory at a time and do not include elements of another array/dimension.
10. The ability to control the storage mapping for a structure, e.g. store each dimension of a multi-dimensional array in its own logical page, container or physical file.
11. The ability to compress and decompress one or more dimensions of a structure with an algorithm supplied by the user.
12. The ability to support algorithms that map the integer “co-ordinates” of an element into other co-ordinate systems. For example, in the mesh example

above, it might be possible for the user to request: Temperature[{40W,30',0"}, {20S,6',0"}] and have Objectivity/DB find element Temperature[40,-20][25,5].

Caveats:

1. Note that this MRD does not require pointer-addressable arrays of objects, but it would be a nice feature if the implementation could support it.
2. Regridding operations would be done by separate frameworks or class libraries, not within Objectivity/DB (hence the constraint in requirement 6 above).

Part of an existing feature or does it require another feature, if so, which one?

- This feature should be an optional feature of the C++, Java, Python and C# APIs.

How is this problem being solved now, and why isn't that acceptable?

1. Users have to write their own methods for handling this kind of structure, adding a mapping layer and increasing storage overheads.
2. It imposes an unnecessary burden on developers and wastes computing and storage resources.

What languages must support this capability?

- C++, Python, Java and .Net for C# in that order of priority.

Which platforms must be supported?

- Linux and Windows.
- Solaris, but only if there is sufficient demand.

Do any commercial competitors already have this feature?

- ObjectStore can support pointer arithmetic on arrays and multi-dimensional arrays.
- Matlab (an application development framework).

Customers who require this feature

- Scientific and engineering users – The JHU SDSS Project had to write their own library for handling nested mesh structures.
- Energy related applications, such as seismic or hydrological surveys and reservoir modeling applications.
- Data fusion applications.
- Complex financial simulations.
- Image processing and management applications.
- Medical equipment applications, e.g. for CAT scans and experiment results.
- Customers building data mining tools for very large datasets.

Revenue at risk, or which could be won

- Implementing this feature could give us new traction at sites such as SLAC, JHU, LANL, CERN, the oil and gas industry, hydrographic and complex (modeling and datamining) financial applications. It would also be useful in data fusion applications.

When is this required?

- Post Release 10, but no later than late 2011.

Additional Notes

It is worth noting that the ability to handle these structures and to exploit existing Objectivity/DB strengths, such as fast navigation, will give us a significant competitive edge over other ODBMSs and all RDBMSs.

We will also need:

- Marketing collateral, including promotional material and a special area on our web site.

- Technical Publications.

- New QA material to prove that the API works and is interoperable with other platform and language combinations.

- Licensing costs are to be determined.

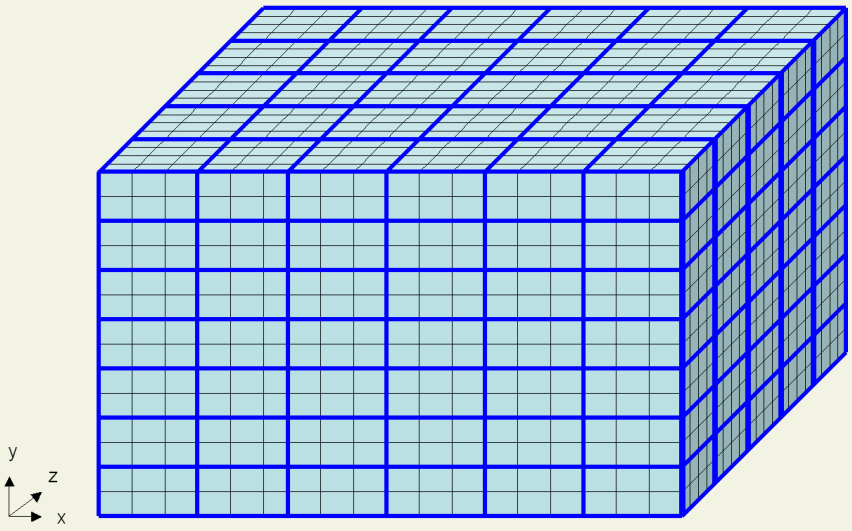
- This feature would be required by any MRD covering SciDB, Matlab or IDL.

References

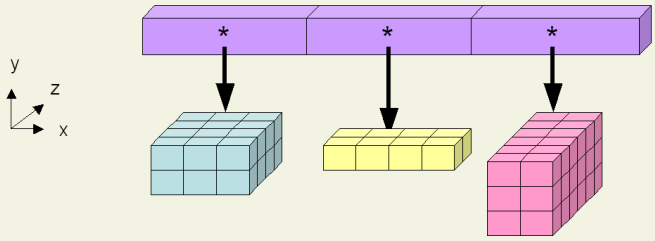
1. Matlab – http://www.mathworks.com/.
2. IDL - http://en.wikipedia.org/wiki/IDL_(programming_language)
3. Finite element method - http://en.wikipedia.org/wiki/Finite_element_method

APPENDIX A – ADVANCED ARRAY TYPES

Multidimensional Array – Using x,y,z this one has dimensions 6 x 7 x 5 x 3 x 2 x 4



Array of Arrays – A 3 element array points at 3x2x4, 4x1x2 and 2x3x6 arrays



Nested Arrays– A 3 element array points at 3x2x4, 4x1x2 and 2x1x6 arrays

Elements of the 3x2x4 point to 2x1x2 and 2x2x4 arrays

An array within the 2x1x6 array points to a 2x1x2 and a 4x2x4 array