LOFAR Data Format revisited

In document LOFAR-DATAFORMAT-001 Sydney Cadot describes the requirements for the LOFAR data format and shows that HDF5 meets about all requirements.
The casacore table system has not been discussed in this document, but meets all requirements with the exception of:

- fully nestable data types
- binding to Matlab and IDL
- installable on Windows

A few requirements mentioned in the document are debatable, especially the one in 3.4.4:
3.4.2. Accomodating 32-bit file systems.
This is not needed anymore. All operating systems support 64-bit file systems nowadays.
3.4.4 The data format should be optimized for sequential access of large arrays.
This should be the opposite, because different applications (flagging, calibration, imaging) require very different access patterns to the same data. Instead the data format should allow for different efficient access patterns.
3.4.5. Pipeline processing.
It is not needed to process data in a sequential way from tape. Possible processing of data in streaming mode will use very different data formats and is outside the scope of a disk-based data format discussion.

Furthermore some possible requirements have been omitted:

- support of a boolean data type (in 3.5.1)
- support of concurrent access by multiple processes
- distributed storage. Note that distributed processing is a separate topic
- thread safety

Needs for visibility data access

The main data axes are baseline, time, and frequency. The application defines in which order the data will be traversed. For example, calibration steps through the data in time order, while for imaging it is preferable to step by frequency channel. Flagging is usually done per baseline in a running time/freq window. Plots can be made in all kinds of ways.
The data can be regular, but that is not always the case. Shorter baselines might use longer time integration.
The file format should be such that data traversal is possible in the various directions. The access in those directions shoyld be about equally fast.

Brief comparison of HDF5 and CasaTables

Both the Tables and the HDF5 data format are well suited for large collections of structured data. They share some characteristics, but differ in many others. The main difference is that HDF5 is hierarchical in nature, while CasaTables is relational.
The 1980's showed a move from hierarchical data bases to relational data bases because the latter offer more flexibility. Hierarchical data bases are (too) hard to traverse in a way different from the hierarchy.

The following table gives a summary of the main differences between the formats and their (dis)advantages.
HDF5 has a much wider user base. Hence, some more tools are available. However, the casapy tools like tablebrowser, tableplot, and casaviewer and the Table Query Language make inspection (and change) of CasaTables very easy.

	HDF5	CasaTables
File structure	Usually single file (can be multiple)	Directory of files Each storage manager is a file
Control how Data are stored	Little (everything in one file)	Each column can be bound to a storage manager best suited Storage managers can be loaded dynamically, so very adaptable
Distributed access	Yes, through MPI-IO	No Individual tables are needed
Data structure	Hierarchical (using groups) - hierarchy defines traversal order Links	Flat (like Relational DB) - easy to make arbitrary selection - easy to traverse in arbitrary order
File size limit	Up to 64 bit file system limits	Up to 64 bit file system limits
Attributes	Yes Set per group and dataset	Yes Hierarchical Keywordset for table and per column
Data Types	All basic types Boolean type as byte Complex types (using compound) Variable length strings Fully nestable compound types N-dim array of all types (also compound) No empty arrays	All basic types Boolean type as bit Native complex types Variable length strings Limited compound types via Records N-dim array of all types No array of compound types
Query, sort	Not available in C++ Available in Python (PyTables)	C++ interface Higher level TaQL (SQL-like) Can also create, update, delete, and insert
Concurrent access	Not supported	Multiple readers and writers supported by means of lock on entire table
Thread safe	Yes, if built so	No
Python	pyhdf5 gives access to hdf5 API Several derived products in Python (e.g. PyTables, pydal)	pyrap gives read and write access to all data pydal
Tools	Several tools e.g. h5dump for a simple dump h5view for view, plot, and edit	Tablebrowser for view, plot, and edit Tableplot for arbitrary xy-plots (part of casapy, not casacore) TaQL
Performance	Depends heavily on data accessed Raw data arrays 40 MB/sec Fast access in all array directions if tiled correctly and cache setup well However, very slow when retrieving smallish data sizes (e.g. lines)	Depends heavily on data accessed Raw data arrays 40 MB/sec Fast access in all array directions if tiled correctly

Array tiling	Chunked (tiled) storage per array Tile cache for per data set (needs reopen to change cache size) No info about cache behaviour h5repack can retile (makes copy)	Tiled storage per array across rows Fully controllable tile cache per array Automatic cache setting for an access pattern Full info about cache behaviour tablecopy can retile
Array slicing	Yes	Yes
Compression	Yes szip, bzip Scaling of float to short	Yes IncrementalStMan (only stores differences) Virtual Storage Managers to scale e.g. float to short
Virtual column	Not supported	A data manager can be virtual (e.g. VirtualTaQLColumn)
Data/Storage Managers	Only HDF5 defined storage managers	Three predefined storage managers Virtual columns New storage managers possible and are automatically loaded as needed
Units and Coordinates	Not supported	Fully supported Units also supported in TaQL
Support	HDF5 support group and forum are very responsive Only serious bug fixing New developments only if paying	Very good local knowledge
Platforms	UNIX, MacOS-X, Windows	UNIX, MacOS-X
Language	C, limited C++, Python, Matlab, IDL	C++, Python
Use	Wide-spread	Limited
Documentation	Quite extensive, but not always clear	Good class documentation Some notes
Robustness	Very robust Very small chance of file corruption in case of machine crash	Very robust Very small chance of file corruption when writing and machine crashes
Disk space usage	Updating compressed datasets can cause waste of disk space	Resizing an array (making bigger) can cause waste of disk space
Data deletion	Attributes can be deleted Datasets cannot be deleted	Columns can be deleted Rows can be deleted depending on storage manager

Examples of CasaTables flexibility:

- Peter Fridman can access the table file containing the DATA directly in his RFI software.
- It is straightforward to store the DATA and FLAG as a normal file (outside table system) and access it later as a table column using a dynamically loaded storage manager (like LofarStMan).
- FLAG can be a virtual column on top of LOFAR_FLAGS which can be an Int or so to have multiple flags per visibility.