NOTE 255 – AIPS++ Table System Revisited

Ger van Diepen, ASTRON Dwingeloo

2003 February 3

1 Introduction

The AIPS++ Table System is based on the table model outlined by Allen Farris in the beginning of 1992. At the start of the AIPS++ project it was decided to develop the Table System as no other package seemed to support the flexibility wanted.

See the old AIPS++ data base document for a description of the very first version of the Table System. Since then the system has been changed and enhanced considerably. The description of the Tables module gives a good overview of the current capabilities.

The Table System is used throughout the AIPS++ project to offer uniform access to all data. The table browser can be used to view all kind of tables. Particular examples of AIPS++ data stored in tables are the MeasurementSets and the Images.

In the remainder of this document the most important properties of the Table System are given. Thereafter the strong and weak points are discussed.

2 Global Features

- The Table System resembles a data base system. It consists of tables containing columns with data of any basic AIPS++ data type (integer, floating point, complex, string). Support for keywords (a la FITS) is provided. Furthermore it supports arrays of all data types (thus not only as blobs). It is, for example, possible to get a slice of an array in the table. The shapes of arrays in a column can be fixed or variable. A column or keyword data type can also be a record making it possible to store AIPS++ `Record` objects into the keywords or column cells. Finally a keyword can contain a reference to another table which is used for subtables of a table (as extensively used in the MeasurementSet).

- Meta data describe the table layout and help to quickly find data. Table System access is based on row number.
  Persistent indices on table data are not supported, but temporary memory based indices make it possible to quickly find the row numbers containing the required data.

- All I/O is done by means of a storage manager layer, so the logical and physical view of a table is well separated. A table can have multiple storage managers, so each column can be stored in the way most suited for the data in that column. Furthermore the Table System supports so-called virtual columns. The data in those columns are not stored but derived on-the-fly from other data. This feature is mainly used to compress columns containing complex data to short integers.
  The tiled storage manager (TSM) is a unique feature of the Table System. It makes it possible to store array data in a tiled way to achieve that access along all axes is about equally fast.
  All storage managers store the data in a canonical format to make access from multiple platforms possible. Boolean values are stored as bits.

- Concurrent access is fully supported. However, only table locks are available, thus no fine-grained page or row locking is possible. See note 256 for more information about locking in the Table System.

- The Table System uses large files where possible, so there is no 2 Gb file limit. On the rare systems not supporting large files the storage managers can be chosen such that each individual file does not exceed 2 Gb.

- Access to the data is very flexible. Addition, change, retrieval, and removal of columns, rows, and keywords can be done at will.

- A powerful query language (see note 199 on TaQL) makes it possible to select arbitrary subsets of a table. TaQL resembles SQL, but lacks joins. However, it supports subqueries and moreover all its functions work on arrays as well.
  It is important to note that the result of a selection and sort is a table in itself which references the original data (i.e. shallow copy). So changing the data in a selection changes the original data. This is

regarded as being very useful. There is a function available to turn a selection into a deep copy.

Note that TaQL can also be used for doing queries on in-memory objects. This is used in the ACSIS system to make a selection from a set of `Record` objects.

- The TableMeasures module makes it possible to store and retrieve Measures transparantly in and from a table.

3 Where are Tables used

As said above the Table System is the main AIPS++ storage mechanism.

3.1 MeasurementSet

The MeasurementSet uses tables to its full extent. It consists of a main table containing the main data. Several subtables contains additional information like the description of antennas, feeds, sources, etc.. Usually the TiledShapeStMan tiled storage manager is used to hold the data in the main table. This storage manager is capable of handling arrays with varying sizes. A tile is 3-dimensional with axes correlation, frequency, and rest (meaning time, baseline, fieldid, etc.). In principle this tiling offers equally fast access when accessing data in frequency direction or in time or baseline direction. However, because the rest axis contains both time and baseline, the access can sometimes be slowish.

3.2 Image and Lattice

An image is stored as a table where the image array is stored using the tiled storage manager. An image can have zero or more masks. Each mask is a boolean array stored in a subtable. Furthermore an image has a log table (for history) which is also stored as a subtable. Other image information (like coordinates) is stored as keywords.

Images are using the so-called lattices which are memory-based or disk-based arrays. Lattices can easily be traversed in any order. The Lattice Expression Language makes it possible to define expressions from lattices. Lattices are commonly used in AIPS++. For instance, the autoflagger uses lattices to assemble statistical information for automatically flagging MeasurementSet data.

3.3 Log Table

All log information of AIPS++ sessions are stored in the log table. In general it works fine, but if the log rate is too high the system cannot cope with it. The main reason is that the log table can be filled from several processes, so it has to extensively use table locking for synchronization. It would be better if a single logging client was running, so the log table had only one process writing into it.

3.4 Miscellaneous

Other places where tables are used are:
- Calibration tables
- Observation catalog
- Sky catalog (used by viewer)
- Tables used by Measures system (IERS tables, etc.)
- Westerbork TMS system

4 Benefits

4.1 Data types and arrays

All standard data types are supported (char, short, integer, float, double, string). Also single and double precision complex data types are supported. A big advantage of the Table System is that it can handle scalars as well as arrays of all thosetypes.
Another benefit is that Measures can be stored in tables.

4.2 Storage Managers

The Table System has some specific data (storage) managers:

- The Incremental Storage Manager can save quite some storage by only writing data when they change. In the main table of the MeasurementSet it is quite heavily used.

- The Tiled Storage Manager is used to store data in a tiled way. It makes it possible to access to image along all axes in an equally fast way (depending on the tile shape). Comparison with a package like Miriad showed that you pay a little penalty for access in the X-direction, but it is much, much faster in the Z-direction.

- The ComplexCompress data manager makes it possible to compress single precision floating point data to short integers to save a factor 2 in storage. It is completely transparant to the application and user, so one still sees the floating point data.

4.3 Concurrent access

The ability of safe concurrent access is very nice. Despite its shortcomings, autolocking makes life very easy, since it frees the user from all locking issues.

4.4 TaQL

TaQL is a very versatile query language with full support of arrays. Users highly appreciate it because it offers easy selection of subsets of a table. Since these subsets are also tables (referencing the original data), one can easily change the data in a subset of a table. TaQL can also be used to sort a table or to sort it uniquely.
The support of arrays makes TaQL in a way superior to SQL. Some selections can be expressed very elegantly in TaQL, while being tedious in SQL. E.g. selecting baselines 0-1, 1-2, 2-3, 3-5 can be done in TaQL as

```
where any(ANTENNA1=[0,1,2,3] && ANTENNA2=[1,2,3,5])
```

TaQL can operate on data arrays in a table, while SQL can only deal with scalar data.

4.5 Glish access

Users greatly appreciate the ease of access via glish. Using the glish functions and tools like TaQL and the table browser it is very easy to examine and inspect the data and change them when needed.
Many users have written scripts to process the data in glish using TaQL and the table access functions in glish.

A widget (`taqlwidget`) exists to form TaQL commands in a query-by-example style.

5 Known Shortcomings

5.1 Access times

Retrieving data from a table is slower than retrieving it from a flat file. It depends on the type and the shape of the data and on the storage manager used. Tests show that retrieving an array from a Table using a Tiled Storage Manager can be about 50% slower than reading it from a raw file when accessing the data sequentially. This degradation in performance has to be weighed against the benefits supported by the Table System as described in the previous section.

Note that measuring IO performance is not as easy as it looks because the UNIX file system keeps small files in memory (small can be as large as 100 Mbytes). One should always do an fsync to be sure the data are flushed to disk.

The main problem seems to be accessing the data in the MeasurentSet in the calibrater or imager. Apart from the 50% degradation discussed above, there can be other reasons for it. One is that the data is usually accessed in a different order than it was stored. Maybe this could be solved partially by storing the data per spectral window. Another thing that might be useful is to use another storage manager than the tiled one.

It is important to note that the TMS system for Westerbork started with use of Sybase for its data bases. They decided, however, to switch to the AIPS++ Table System because it performed much better than Sybase for their particular application.
No comparisons have been made between the Table System and a data base system like MySQL or PostgreSQL. Doing comparisons may be hard because performance will usually be highly application dependent.

5.2 Non-blocking IO

The storage managers only use simple synchronous I/O, though they use a cache to buffer often used data. Non-blocking I/O nor file mapping is used. Especially non-blocking I/O might improve performance, but it requires extra buffer space. It might also be needed to provide more functionality for giving hints about access patterns.

5.3 Robustness

In the early days the Table System was not very robust, but it has improved over the years. In occasional circumstances the data in a table can be corrupted. This could be the case when the system crashes while a critical data portion is written into the table. However, in practice it hardly ever happens because especially in the now commonly used StandardStMan storage manager quite some attention has been paid to robustness.

5.4 Standards

The Table System is not a commonly used piece of software as, for example, MySQL is. It means that only a Glish binding is available. However, it would be not too much work to make, say, a Python binding.

5.5 Locking

Table locking can be problematic if not done properly.
Locking can only be done on the entire table, so for full multi user access a lock should be held as short as possible. On the other hand releasing a lock means that the buffers needs to be flushed to disk, so the extra IO involved could mean that one wants to hold the lock as long as possible.
The AutoLocking mode was invented to solve this problem. However, it has the drawback that it may take a few seconds before the system detects that the AutoLock should be released. This is especially the case in a glish client which might be idle for some time. It is now possible to define the AutoLock inspect time in aipsrc, so one can set it as short as needed.

5.6 TaQL

Although TaQL supersedes SQL with its array capabilities, it lacks several nice SQL features. The most imporatant are:

- joins
- GROUP BY and HAVING
- calculated columns in SELECT

6 Future

The current Table System shows some shortcomings, especially in the area of parallel data IO. This area will get more and more important. Current instruments can already produce lots of data, but future instruments like ALMA and LOFAR will produce even much more.

Because the I/O in the Table System is done in a separate layer, it should in principle be possible to enhance it with other forms of I/O like parallel and networked I/O. However, other high performance data I/O libraries exist and it should be studied if it is better to use such a library. HDF5 is such a library. It has an C++ and Java interface and a viewer written in Java. It supports array tiling (chunking in their terminology).

6.1 Data Storage

Before deciding on how to proceed with a data system the requirements should be clear. It is clear that (near) future systems have to process observations possibly consisting of several terabytes of data. Full support of parallel processing and I/O are needed, but it is the question how.

The optimal way of parallel processing is that the processing is done on the host where the data are. Thus a client should only ask for the result and not for the data themselves. Of course, this is not always possible. It should also be possible for a client to get file data from another node, thus parallel and networked I/O is needed. Parallel I/O will be very important when data are stored in a SAN.

An example might be an image of 4000*4000*4000 float pixels. Such an image might be tiled and spread over several nodes. Many operations (like finding minimum/maximum, adding images) can be done locally on those nodes. So the data system and processing system have to collaborate for optimal performance.
Another example is the selfcal of a very large MeasurementSet. The predict can be done on the node where the data resides and only the residuals and derivatives are sent to the solver. Sometimes, it might even be possible that a solve is done locally.

Data bases are not useful for such data. They lack the ability to deal with large arrays of data or to access chunks of them.

6.2 Administrative tables

Several other tables (log tables, observation catalogs) can be handled with standard data base technology. Public domain packages like MySQL or PostgreSQL should be evaluated. It might be needed to add special indices to deal with queries like a conus in the sky, although it is expected that such functionality already exists.

In principle it is possible to make a storage manager in the Table System that uses a data base system. It is the question whether that is worthwhile. The advantage would be that TaQL could still be used to have support for data arrays in queries.

6.3 Archived Data

Some institutes (e.g. Westerbork) have archived MeasurementSets. They are archived in `ms2archive` format. When support of the Table System is stopped, a tool should be provided to retrieve these archived MeasurementSets in the new data format.