STAR Note: 387 Revision A.
The STAR Tag Database for MDC2

Dave Zimmerman

February 26, 1999

This document is an attempt to answer the questions:
What is the Tag database?
How does someone interact with it during an analysis?
Although I attempt to be clear throughout this note in some cases I have lapsed into jargon. At the back of the note I have included a glossary of some terms which may not be familiar to the reader.

At the outset I would like to briefly describe the STAR tag database prior to launching into more detail. The STAR tag database is essentially a column-wise ntuple which has sets of variables necessary to identify an event (the object which is "tagged") and some set of floats and integers (the tag data) which are suitable for analysis and simple selection. In the section 7 of this note I present some examples of analysis scenarios involving the tag database. These scenarios should help to clarify the use of the tag database tool in STAR analysis.

Certain aspects of the tag database, for example the QATags are not yet fully implemented. In the sections where I am describing post-MDC2 features, I will try to explicitly mention this.

One point that I wish to stress in this note is that the goal of the tag database is not to restrict physicists to a particular analysis style. The traditional methods are still available in the STAR framework. The tag database is a tool which should facilitate certain analyses styles and hopefully open up new possibilities.

1 Introduction

The STAR experiment presents a unique challenge to the traditional data analysis framework. The raw data rate of 1 Hz and a typical event volume of 20 Mbytes will lead to 200 Tbytes stored in the RCF in a given year. In addition to the 200 TB of raw data there will be about 70 TB of associated reconstruction and post-reconstruction data which will also require storage in the RCF. For more details on the specific size of the various pieces of the components see the computing requirements [1] document.

This large data volume requires distribution of events over both disk and tape. Since more than 90% of the data in a running year will be in robotic tape storage there is a premium on efficient access to the data store.

The purpose of the tag database (tagdb) is to optimise access to the complete event store by allowing the physicist to select events based on a "physics index" over the events. The selective access is based on the physics quantities which make up the physics information in the tagdb.

Thus, a physicist who knows that he is only interested in analysing events which have a particular trigger type and a good reconstruction status word (e.g. Trig=10001 and Reco =1) will be able to preselect the identifiers for the events of interest before having to read any data from the tape storage.

Because the tag data base is a data access tool it must be small enough to be permanently disk resident for AT LEAST one full running year of data. The tag database will be analysed and re-analysed in the course of designing an event selection. For this type of access one should try to keep the total volume down in order to minimise the IO time spent reading it.

Given these conditions, a back of the envelope estimation of the maximum size of the tag for each event can be made. There will be 10 million STAR events recorded each year. As an example, in order to keep the total Tag database size to an easily analysed and managed size of 10 Gbytes there can be no more than 1 kilobyte of tag per event.

2 Star Event Store and the Persistent event model

In order to improve efficiency of data access for physics analysis STAR will distribute events across multiple files. For example, for a particular run there will be a distinct RAW data file, a file containing the TPC hits, a file containing the reconstructed track information, a file containing vertex information, as well as other files containing additional components of the events in that run. The components of an event within this run will be distributed across the files for that run.

There are a number of advantages gained by distributing the events across files in this way. Analyses which only require access to certain components of the events no longer need to read complete events onto disk. A physicist is now in a position to specify which of the components of a particular event (e.g. tracks and vertices, but not the hits) are required for her analysis, and to read only those components. Furthermore, the number of events contained in a file (for a particular component) is maximised so the total number of files of each type accessed is smaller, minimising the number of tape mounts necessary for an analysis.

The central component for each event is its event header. Each event has its own event header which contains references to the event's components. The event header is the object one accesses to access each of the various event components.

2.1 Where the tag data base fits into the overall model

The tag database is made up of individual tags which are associated with each event. These tags are referenced from the event header and are themselves a component of the event. Their most important feature as event components is that they are disk resident for the entire running year.

During MDC2 the tag database pieces of a particular event will be single structs per group per event. For the final implementation, each PWG will be responsible to specify the tag objects necessary for its pieces of the tagdb. (An example of a schema such as could be used for the tag database is contained in the appendix to this note.)

Each PWG's tag object for an event will have its own QATag object. For example, there will be a single QATag for the object corresponding to the Strangeness Reconstruction tag, and one (or possibly several) QATags for the Strangeness MicroDST object(s).

3 Structure of the tag data base

The structure of an event tag consists of two basic components, the "event building" piece and the queryable information. Any object which contains these two components is functionally an event tag. Thus any physicist can construct his or her own tagdb which is optimised for his or her own purpose. The principle differences between a customised tagdb of this sort and the "official STAR" tagdb are twofold:

- The STAR tagdb is managed by the Physics working group coordinator and the STAR database administrator
- A subset of the information in the STAR tagdb is used to build the Grand Challenge index. This index determines which quantities are available for Grand Challenge range queries.

In the following sub-sections I will describe the makeup of the "event building" and queryable information features of the tagdb.

3.1 The event building pieces

As described above, STAR events consist of disk resident event headers and components which are distributed through a number of files in tape storage. In order to uniquely specify a particular event the tagdb entry for each event must contain an event identifier (in STAR this is an Objectivity object identifier (OID) to the event header) and a file identifier (fileID) for each file, necessary to build the events. In STAR the file identifiers can be looked up in the FileCatalog which consists of a fileID to UNIX-style pathname mapping for the files in the STAR event store.

Technical Comment: Strictly speaking the file identifiers can be retrieved by navigating to the event collection from each event header, but the file information is kept in the tag for ease of index building.

3.2 The Queryable Physics Information

In order to function as a useful index over the event store the tagdb contains a set of integers and floats in an ntuple-like form. These variables are used to make selections (also known as queries) to pick out events of particular interest for a given analysis. The "Queryable Physics Information" corresponds to the information which can be used to form the selections. Appendix 1 contains an example of the sort of information which might be used to make up the queryable part of the tag. **The principle feature of this information is that it should be selected not for physics analysis or as a repository for event summary information, but as query quantities.** Thus run number, event number, trigger word, status bits will certainly be included. Event summary information such as histograms are not appropriate elements of the STAR tagdb. Since data such as histograms are comparatively high volume and do not lend themselves directly to querying they should be stored in an event summary component of the event.

There are many physics quantities which are useful to pre-select the events which will be accessed for a given analysis. Some examples of interesting query selection criteria are:

- Track multiplicity
- Pt of the highest Pt track.
- Multiplicities of strange particles.
- Number of EMC clusters above a certain threshold

3.2.1 Comments on queryable information

As a final comment on the queryable physics information I'd like to make Two additional comments.

It should be pointed out that the tagdb information isn't*necessarily* physics analysis information. If one discovers that there is some variable which correlates well with on to the set of events one wants to analyse that would be a good candidate to be an event tag **whether or not that variable has a clear physics meaning**.

Lastly, I'd like to stress that while the tag database (in conjunction with the Grand Challenge software) is a powerful analysis tool it cannot replace the need to carefully control and manage systematic errors arising from bias and data calibration concerns. An analysis using the tag database for event selection requires as much, but not more, diligence in order to extract a physics measurement from an event sample.

3.3 Structure of the STAR tags

This section describes the structure of the STAR tagdb in both its Objectivity and Root forms.

It is worth commenting on the existence on of both a Root and an Objectivity tag database. Objectivity has been the principle development tool for the STAR tag database. In winter 1998 it was decided that STAR would adopt Root as the analysis framework. In the course of developing Root tools for STAR, it became evident that the Root Tree form (a Tree in Root is a generalised form of the PAW ntuple) is particularly well-suited to the tag database problem. The flexibility and tight integration into the analysis environment which Root offers are very useful for a tag database tool.

During MDC2, STAR will have both an Objectivity and a Root tag database, which will have identical information (at least at Reconstruction tag creation time). After MDC2, the official STAR tag database will exist only as a Root tool.

3.3.1 The ObjectivityDB tag database

The Objectivity tag database (ootagdb) schema is described in an Appendix to this note. The important distinguishing features of the ootagdb are that its schema is pre-specified and unchangeable before the creation of the federation. This decision was taken in order to avoid the problems associated with schema evolution of the STAR Objectivity federation.

In MDC2 the Objectivity tag database will be used to build the grand challenge index. After MDC2 the root tag tree will be used as the source of data for the grand challenge index.

3.3.2 The Root Tag Tree

The root tag tree has similar schema to the ntuple-like Objectivity tag data base. In particular it has the information necessary to identify an event header and some set of floats and integers which are suitable for analysis and simple querying.

The Tree consists of a number of distinct branches. Each branch is associated with a particular physics working group or specific tag function:

- The Tag Administration branch contains quantities such as event identifiers, associations to QATags, processing status words and other information necessary to the functioning of the tagdb.
- The PWG coordinators' portion of the tagdb consists of a set of branches which contain information which is relevant to the entire STAR experiment (trigger words, experiment status) for each processing stage. These branches will be writable only by the PWGC. PWGs may request that information be added to the PWGC branches but the updates must be approved by the collaboration.
- The strangeness branch contains tags requested by the strangeness PWG.
- The Event-by-Event branch contains tags requested by the EbyE PWG.
- Other branches as necessary will be available for each of the other PWGs in the STAR collaboration. I have not listed all of them here only to save space.

Each of the PWG branches will be writable and manageable by the individual PWGs. Because of the limited amount of disk space available for tags the PWG branches may not be permanently disk resident. This question requires further study and experience. Note that there will certainly be some overlap in the tag quantities among the various PWGs. Tags which are useful to more than one PWG can be promoted from PWG tags to the branches of the tag tree managed by the PWGC. This process will require approval of the PWGC.

3.3.3 Quality Assurance Tags, Random Tags and Status bits

The QATag object is a component which is being introduced for the first time in MDC2. There remains some work to integrate it with the STAR environment. For MDC2 it will be available, but it is not yet fully functional.

Associated with each tag object there is a quality assurance object which contains character information about the tags contained in that branch. As information is added or updated in each branch, new QATags are created. These include both character string and time stamp data. Each tag has a reference to its QATag object. The QATags enable a user to keep track of the conditions under which the tag information for this tree was created.

For example, when there is a re-evaluation of the quantities which make up the high-Pt reconstruction tag one would update the values in the tag and the QATag. So one would:

- Invoke a set method on a value in the data base.
- Create a new QATag object with an appropriate comment and time stamp.
- Create a reference from the Tag object to its QA object.

In the TagAdmin branch of the tag data base there is a variable called Tagrand, whose value is a random number between 0 and 2000. This tag can be used to select randomised subsets of the events required for a particular analysis. Thus, if one wishes to make a fast turnaround analysis which samples a random subset of the events of a certain type, one can make a query using the Tagrand variable. (e.g. MyQuery .eq.True .and. Tagrand .lt.2 will select only .1% of the events for which MyQuery is true.)

In the TagAdmin branch there is also a set of Status Bits which can be used to indicate detector state, reconstruction status and other relevant information. At this time 16, bits are present in the schema. After MDC2, requests for more bits will be accepted.

4 The Life cycle of the Tagdb

For each event an entry is created in the Objectivity tagdb at the time the event header is created. The schema of the tagdb is initially filled during the event reconstruction, but the values in the schema are available for updating throughout the processing of the event. For example, if after the physics analysis groups have had a chance to refine their algorithms the value of a tag (e.g. the number of Kaons in a particular event) changes the value of that tag can be changed. The procedure for changing the value of a tag is described below.

Note that the Objectivity tag database will probably be fully replaced after MDC2 by the root tagdb (or tagtree). For MDC2 STAR will have both a root and an Objectivity version, in the following discussion I will concentrate on the root tagtree as that is of more utility and long term interest.

4.1 The mechanics of the Root tag

As each tag is created and updated, references are added to a QATag for this particular tag object. In figure 4.1 the growth of the tag database is illustrated.

4.2 Personal tag trees

One of the most powerful aspects of the root tag tree is the ability for an individual physicist to copy the events tags to his or her own personal root tree. At its creation this copy will still contain some subset (or all) of the queryable information from the root tree as well as the event retrieval information from the tag. Thus the personal copy can act as a customised tagdb suitable for analysis at one's home institute.

When one has a personal copy of the tag database one can easily customise it to one's own needs. This will enable creative development of new tags which can be introduced to the set of PWG tags.

5 Interacting with the tagdata base

The principle analysis goal when interacting with the tag database is to determine which events are good candidates for one's particular analysis. Using the tag database and the grand challenge architecture, one can identify events of interest and create a list (or event collection) This event list can be passed to the grand challenge software as a "query by collection". In this section I will describe the a scenario of the expected use of the tag data base.

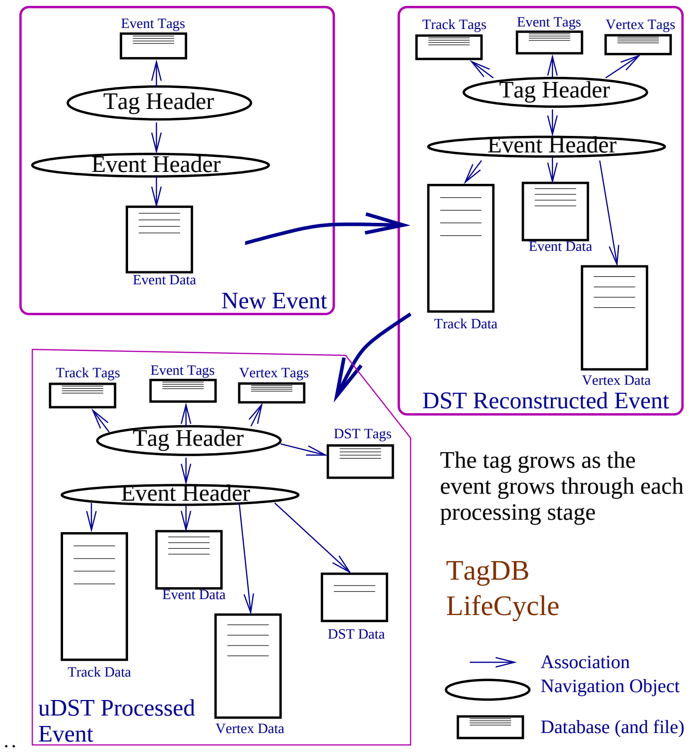

Figure 1: This figure shows the growth of the tagdb as the processing of an event is carried out. As each processing stage occurs a new branch is added to each PWGC portion of the tagdb.

5.1 Updating or adding information in the tagdb

The initial state of the tagdb is set when an event is first created in Objectivity. If further processing reveals that information in the tag requires updating there are standard procedures to follow:

1. Determine that the new or updated tag information is appropriate, possibly by adding the tag to one's personal Root Tree
2. Ask the PWG coordinator for your PWG to add the information to the appropriate branch for your group.
3. Create a QA tag describing the update and add a reference to this QATag from the branch you are using.
4. If the information you are adding should be added to the grand challenge index you should ask the tagdb admin to set the updatable flag.

6 Example of Using the Tag Database

The principle utility of the tag database is in the creation of the event lists of events of interest. In the following I will illustrate the use of the tag database with the grand challenge software.

6.1 Schematic Example:

This example describes a series of steps which go into making an analysis based on the tag database and the Grand Challenge software.

6.1.1 Initial guess for a query

As a first stage in designing an event query one can make a general guess as to a query. For example, as a first pass one may select only based on a trigger condition. The first stage query attempt should be tested by performing a grand challenge range query estimation. The GC estimate will tell you the number of events and bytes that will be retrieved, and estimate the total required time needed to satisfy your query. If this number is inconsistent with your intuition or goals, it may indicate that you should further expand or refine your query.

6.1.2 Building an event list

In order to refine one's query, the best tool is the Root TreeViewer tool. This graphical tool allows easy query design and histogramming of the variables in the tag tree. Once you have a satisfactory query you can use the TreeViewer to output a root-style eventlist. In order to convert from the root event list to STAR event header identifiers (Objectivity OIDs), you must re-read the tag tree, but this is easily done.

Armed with your list of OIDs for events of interest, you can present the list to the Query Estimator and receive a new query estimate. At this point one might wish to execute the query and begin the analysis of the interesting events. Otherwise, if one is still developing the analysis, further refinement of the query, or event sampling using the Tagrand might be preferable before launching on a long analysis.

6.1.3 Evaluating the tag information

Relying on the tag database as an query and data access tool in no way relieves the physicist performing the analysis from careful evaluation of his event sample. The physicist must carefully check run and reconstruction information to confirm that the events in the analysis sample are as appropriate to his or her analysis. This work is abetted by the QATags and the calibration database.

The tag database does assist in this process by having references from the tags to QATags which describe the data. The QATags and the tag status bits for an event are the primary means of determining the state of the information in the tags.

6.1.4 Once your query has been finalised

Once you have specified your event list you can pass it to the grand challenge as a query by collection and begin execution. The event list should be maintained as a record of your query. The event list can be important not only as a book keeping device, but in order to facilitate comparisons of event lists with ones colleagues.

7 Glossary of terms

Collection: A Set of associated events. e.g. A given STAR run is a collection. The set of My QGP candidates can be a collection.
Federation: An Objectivity specific term which means, in essence, the set of Objectivity databases for a particular project. The STAR Objectivity event store is a single federation. It contains many different databases.
Grand Challenge Software: Information on the GC can be found on the web [2]
Grand Challenge Index: An index which contains some physics information (read from the tagdb) and event identifiers (OIDs). The Grand Challenge Software uses this index to determine which events satisfy a given range query.
Grand Challenge Range Query: A range query is an event selection. For example, give me all events which contain more than 5000 pions is a range query. Range queries can be expressed mathematically if (NumberofPions $>$ 5000) = TRUE
Grand Challenge Query by collection: The Grand Challenge Software can accept collections as queries. This amounts to passing the Query Estimator/Executer a list of events. This type of query is called a query by collection.
Objectivity: An object oriented database which STAR uses as part of its event store.
OID: stands for object identifier. An Objectivity term which uniquely specifies an event. In STAR, events are uniquely identified by the OID of their event header.
Private Tag database: A copy containing some fraction of the tag database which individual physicists can use in developing their own customised analyses. By creating one's own tag database one is freer to introduce new tags specific to one's own analysis without the overhead incurred by adding tags to the official STAR tagdb.
Quality Assurance Tag: An object which is used to describe the information in a tag. An example of a QATag would be an object which contained a time stamp of the time that the tag in question was created and a a character string with information like:
QATag$-$ $>$AddString("2000 Reconstruction pass 2. Using Ullrich Lambda definition.")
Query Estimation: A feature of the grand challenge which allows one to learn how many events satisfy a given query, how many bytes will be returned and an estimate of how long it will take to satisfy your query. This extremely useful feature will help to keep people from mistakenly requesting all the data from every event in the STAR event store or other query errors. It is especially nice to use as feedback for query design. For example, if I wish to access low multiplicity events, which I know make up about 1% of the events in the store I could try a query like: Give me all events with less than 10 tracks. If my query estimation determined that I would access 10% of the events in the event store I might decide to refine my query a bit by requesting: Give me all events with less than 10 tracks and for which the primary vertex is well centred in the detector. Note that, if you are doing a query by collection the answer will be the number of events in your collection.
QATag: A Quality Assurance tag.
Query: A selection. Give me all events with a track with more than 15 GeV of transverse momentum in them is a query.
Root: An analysis framework which STAR will use. More information can be found on the web

at URL: root.cern.ch
Root Tree: A more powerful version of the familiar PAW column-wise ntuple.
Schema: The structure of objects in a database is the database's schema. In general the schema is really the set of all header files describing the objects. An example of a schema for a Tag object is found in the appendix to this note.
Schema Evolution: Changing the schema of a persistent object. As one might imagine, changing the structure of objects stored in a database is a fairly complicated and difficult task.
tagdb: The tag database.

8 Appendix: An example schema for a tagdb

In figure 8 there is an example of a schema which is used in a test tag database at LBL.

```
//////////////////////////////////////////////////////////////
class EvTag : public ooObj {
private:
  d_Long _N_Ev;                 // Event Number
  d_Long _N_Rn;                 // Run Number
  d_Long _num_Pion_p;           // Number of positive pions
  d_Long _num_Pion_n;           // Number of negative pions
  d_Long _num_Pion_zero;        // Number of neutral pions
  d_Long _num_Kaon_p;           // Number of positive Kaons
  d_Long _num_Kaon_n;           // Number of - Kaons
  d_Float _tot_trans_Mass;      // Sum of all transverse masses
  d_Float _SumofMomentum_x;     // Sum of all px in detector
  d_Float _SumofMomentum_y;     // Sum of all py in detector
  d_Float _SumofMomentum_z;     // Sum of all pz in detector
  d_Long _nTags;                // number of tags
  EvOID _EventID;               // Struct for Event ID FD-DB-PG-SL
  d_Short _fid;         // file identifier of event body
  d_Short _fidaux;      // file identifier of file aux
  d_Short _fidcomp1;    // file identifier of file for component1
  d_Short _fidcomp2;    // file identifier of file for component2
  d_Short _fidcomp3;    // file identifier of file for component3
  d_Short _fidcomp4;    // file identifier of file for component4

  d_Float _TagRand;     // random used for tagging

  ooRef(oParticles) _toEvComp_Tag;
  ooRef(EvHdr) _toEvHdr_Tag;
public:
  // Fill the tag data base Methods
  ooStatus PutLong (char* TagName, d_Long TagVal) ;
  ooStatus PutUShort (char* TagName, d_UShort TagVal) ;
  ooStatus PutFloat(char* TagName, d_Float TagVal) ;
  ooStatus PutEvOID()  ;
  ooStatus PutEvRef(ooHandle(oParticles)& oPartH)  ;
  ooRef(oParticles) GetEvRef();
  ooStatus PutHdrRef(ooHandle(EvHdr)& EvHdrH)  ;
  ooRef(EvHdr) GetHdrRef();
  d_Float GetFloat(char* TagName) ;
  EvOID GetEvOID()  ;
  d_UShort GetUShort (char* TagName) ;
  d_Long GetLong (char* TagName) ;
};
```

..

9 Bibliography

References

[1] http://rsgi01.rhic.bnl.gov:80/STAR/html/comp_l/ofl/reqmts9708/

[2] http://www-rnc.lbl.gov/GC/

[3] http://rsgi01.rhic.bnl.gov:80/STAR/html/comp_l/dataproc/