1999-11-18

Thoughts on Hall D Database Use

Elliott Wolin
16-Nov-1999

During my talk on Online Strategies at the recent Hall D DAQ/Trigger Workfest I briefly discussed the use of databases in Hall D. Below I elaborate on this important topic.

Why Use Databases

Databases help to solve two general problems. From the top-down perspective, they help you manage complexity; e.g. tracking the analysis state of the large number of files produced, indexing the numerous sets of calibration constants and snapshots of the detector configuration, etc.

From the bottoms-up perspective, databases separate the action of specifying what data you want from the mechanics of actually retrieving the data from the datastore.

Database Choice

There are many types of databases we should consider, including relational (RDBMS), object-oriented (OODBMS), file-system based, and custom databases. I estimate we have 2 years before we need to make final decisions. Which one(s) we choose depends on:

1) what we intend to do with them
2) how mature the technology is
3) how much manpower is required to develop and maintain them

Concering point 1, a major question is whether to store the online event data in a database, as is done by BaBar, LHC experiments, etc. I note that in their case a small number of interesting events are embedded in a large number of uninteresting events. In our case, all hadronic events are interesting.

Less controversial is the use of databases for online configuration and offline calibration data, as this is routinely done.

Concerning point 2, object-oriented database systems are developing rapidly, and the state of the art might be well advanced (due to the efforts of BaBar, RD45, etc.) in two years. Relational database systems are mature already, and are in widespread use.

Concerning point 3, our guiding principle should be that our choices reflect the scale of the experiment. The Hall D collaboration is much smaller than the large HENP collaborations (BaBar, STAR, CDF, D0, LHC experiments, etc); we can only afford to push the state of the art in a few carefully chosen areas.

My Prejudices

I list my prejudices to create a starting point for further discussions:

1. We should use databases

Aside from the raw data (discussed below), the days of storing everything in flat files or custom databases are over; we should use the standard technology, relational or object -oriented, as appropriate. The database might point to flat files, but our thinking should be (in Chip Watson's words) "database -centric" from the start.

2. Don't store raw data in OO databases

I don't think we should store our online data in an OO database. I predict the state of the art will not advance far enough in 2 years to allow us to devote the required manpower or money (OO databases probably will still be fairly expensive).

I also don't think OO databases are particularly suited for our raw data. We use a loose trigger; i.e. our events are "min -bias" events.

After reconstruction an OO database might be appropriate, as our events can then be classified by final state. Then the notion of iterating over a special subset of events becomes useful. Will our online farms perform final reconstruction...?

We might consider using ROOT (a kind of OO database) to store the data, as well as CODA format, or a custom format.

3. Non-event data

Relational databases are mature, widely available, cheap (MySql is free), well understood, simple to use, and easy to interface to (c++, c, perl, java, etc.). They are well suited to holding online configuration data, calibration constants, logbooks, etc, and are in widespread use.

OO databases support more complicated data structures, and could work well for non-event data. Relational database structures (tables) are easily mapped into objects in an OO database. Access to OODBMS's is more difficult and fewer access methods are available (perhaps just c++, although this might change).

I think relational databases should be our nominal choice, but we should keep abreast of developments in the OO database world.