# Data Base : RCS (Hall A?) online

JLab, Dec 2001

E.Chudakov, JLab

Hall A online etc would benefit from an elaborate, running all the time DB (like Hall B and others do). One can store various information and be able to retrieve the needed one later, more or less automatically:

- Data, nearly stable during a running period
  - Electronics configurations
  - ...
- Data, which may change every day
  - Spectrometer positions, momenta
  - Beam energy
  - ...
- Data, which may change every hour
  - RUN info
  - Pedestals, calibration factors
  - ...
- Data, which may change every minute
  - Various variables read from EPICS
  - ...

Joe Mitchell worked a bit on DB for Hall A, but did not release anything.

# Plans

I started working on it recently (about a month ago). Oririginal intentions: RCS calorimeter information. Now, my plans are:

- For the RCS period provide a pilot DB, storing:
  - Electronics configuration tables (Bob's request)
  - HV for RCS (from HV control program)
  - RCS calibration data (from the analysis programs)
  - RUN tables (from RUN start/end scripts)
  - Some EPICS (from scripts)
- After RCS reconsider the design and add:
  - General calibration features (like Hall B)
  - ...

Selection of the DB type, general:

- Relational DB
- SQL
- Free, open source
- Big enough

Typical interfaces available for the existing DB include: Graphical/commandline front-ends, C/C++, Java, Perl, Tcl/Tk, Python etc.

# Demonstration

The electronics tables, required by the new analyzer, have been implemented.

- List of crates
- Positions of the crates, time dependent
- List of modules
- Positions of the modules in the crates, time dependent

The positions' tables include the starting time for the given configuration (row). This is enough to find the correct configuration for any given time using SQL code:

```
--  Select all the modules for a time 'mytyme'(say, 2001-12-05 21:14:02-0500)
---     in the crate mycrate
SELECT
       tmp3.roc,tmp1.slot,ml.model,mt.type,mt.nchan,tmp1.daqid,
       tmp1.crate_list_id,tmp1.timeentry
FROM
       (  SELECT DISTINCT ON (module_list_id) *
          FROM module_pos
          WHERE timestart< :mytime
          ORDER BY module_list_id,timestart DESC)
       AS tmp1,
       module_list ml,
       module_type mt,
       (  SELECT *
          FROM
          ( SELECT DISTINCT ON (crate_list_id) *
            FROM crate_pos
            WHERE timestart< :mytime
            ORDER BY crate_list_id,timestart DESC)
          AS tmp2
          WHERE tmp2.codastream= :mycoda)
       AS tmp3
WHERE
       tmp1.crate_list_id= :mycrate
       AND tmp1.crate_list_id= tmp3.crate_list_id
       AND ml.id=tmp1.module_list_id
       AND mt.model=ml.model
ORDER BY tmp1.slot;
```

# Choice of the DB

| DB | MySQL | PostgreSQL |
|---|---|---|
| Open source | yes (GPL) | yes (Berkeley) |
| speed | very fast | slower ($\sim 1/3$) |
| SQL | rather limited | rather full |
| documentation | good | good |
| experience | good | good |

At the moment, speed does not look as an important limitation. I selected PostgreSQL for the pilot run because of the following advantages:

- Support of arrays (flexible length) as a column element
- Sub-queries supported
- Transactions supported
- Binary data retrieval

I offer a C intefrace to retrieve the data.