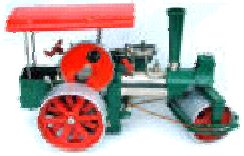
Conditions Database

Workshop Summary

Andrea Valassi

(CERN IT-DB)

http://lcgapp.cern.ch/project/CondDB/



and

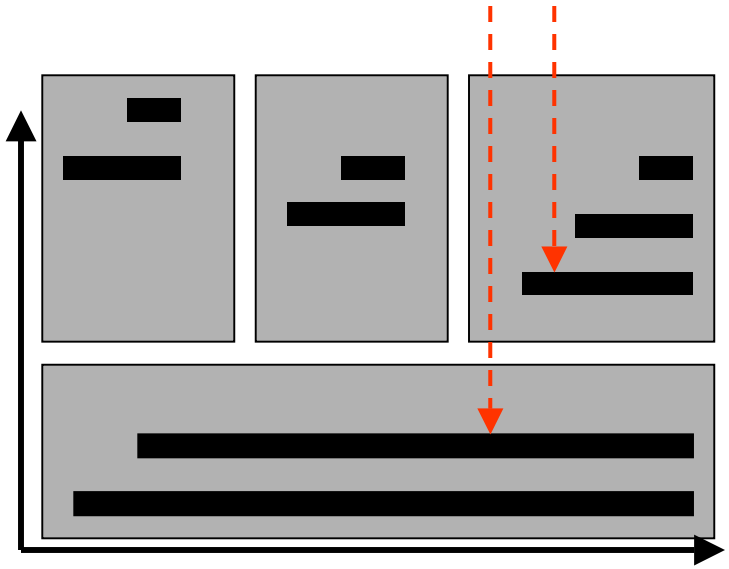
- **Online**
- **Relational data**
  - Data that can be plotted
  - Data stored close to metadata
- **Time stamp**
  - Data stored when values change
- **Measured data**
  - Only one version
- **Unfiltered data volumes**

- **Offline**
- **Object data**
  - Data that can be referenced
  - Data accessible in unmediated way
- **Time interval**
  - Data stored periodically
- **Computed data**
  - Versioning and tagging
- **Filtered data volumes**

Take into account sometimes opposite points of view

Partitioning

- Partitioning by folder
- Partitioning by time
- Partitioning by insertion time
- 2D partitioning (à la BaBar)



Tools

- **Tools to slice/export the data for distribution purposes**
  - Distribution to user laptops or to Grid production centers
  - Exporting a DB referencing external files should export them too
- **Tools to browse the data**
  - Web accessible
  - With editing capabilities: slicing, tagging, retagging
  - With plotting capabilities: history plots

Data item: folder name & beyond

- A single string in the "common" API
  - *"/SlowControl/Ecal/Module1"*
- Use an arbitrary number of relational keys instead?
  - *"type=Slow,det=Ecal,module=1"*

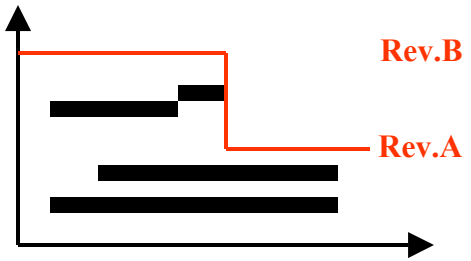
This is the idea behind the Lisbon "CondDBTable with ID"

Version number & beyond

- Use insertion time instead?
  - Examples: BaBar, Compass
- No intrinsic meaning to the version number
- Is it necessary to insert fake intervals to complete HEAD?

Tagging: HEAD & beyond

- **Associate metadata attribute(s?) to computed data**
  - Tag HEAD on subset matching required attributes [David M.]
  - Metadata useful for computed data (algorithm, source…) [Martin L.]
    - But should go inside data content if not used for versioning?
- **Tag a past HEAD (BaBar "revision")**
  - More meaningful if insertion time is used instead of version number
- **Tag different "revisions" in different ranges ("BaBar view")**
  - Easier when using time range partitioning



Time

- **Time interval vs time stamp**
  - Is it not enough to store interval [t0, t0]? Normal folder, many holes?
- **Time vs (run#,event#)**
  - Is it not enough to make the translation in the experiment framework?
  - Need a (set of) variable(s) with strict order relation anyway…

Personal opinion: can be solved by conventions on top of API (no need to modify the API)

Synchronization

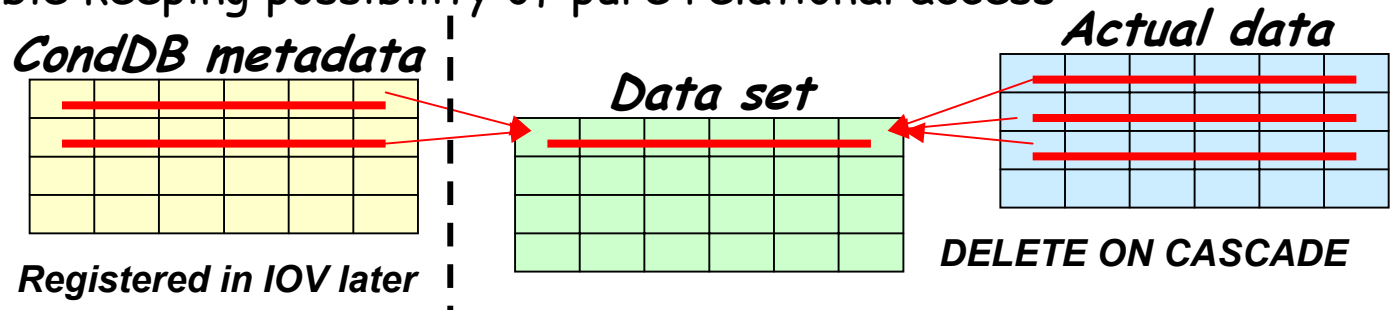
- **Common problem: load condition data for given event time**
  - For processing by offline framework
  - Could develop a common solution if required

NB I assume loose coupling between stored event data and condition data (situation is different if condition data pointer is stored in each event)

- **Also need a memory-resident implementation (cache)**
  - For processing by online farms [Clara G.]
  - Could develop a common solution if required

Stored data: BLOB & beyond

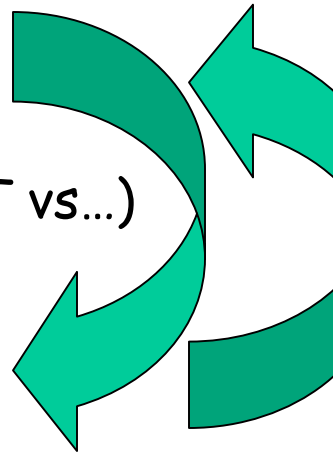
- **Relational substructure of condition objects**
  - Delegate to POOL relational backend
    - If possible, superimpose C++ layer on top of existing relational model
    - Rather than force relational model to follow the needs of the API
  - Encapsulate relational data with same validity/version as a single entity
    - One CondDBObject -> One POOL reference
    - One POOL reference may refer to many rows (of many tables?)
    - If possible keeping possibility of pure relational access



- **Non trivial use cases**
  - History plotting
  - "You notice a problem at time t0. Go forward and backward in time to determine how long the problem lasted for." [Lorne L.]

Boundaries with experiment s/w?

- **Common project**
  - Time/version metadata storage
    - Generic technologies and tools built around that (with plugins)
  - Synchronisation?
  - Not aware of "calibration","slow-control","alignment" concepts…?
    - Only aware of generic technologies (only at higher level: POOL?)
  - Service deployment?
  - …
- **Experiment**
  - Choice of main technology/ies (Oracle vs MySQL vs ROOT vs…)
  - Integration with experiment C++ framework
  - Coherent procedures among subdetectors
    - Classification of data items
    - Choice of time axis
    - Format and data model of actual condition data
  - Distributed computing model?
  - …



Work plan items (1)

- **Release Oracle/MySQL packages in LCG infrastructure**
  - SCRAM and LCG CVS repository ***(but it must be gcc 2.95.2…)***
  - Now: in the state they are now (different APIs and tests/examples)
  - Then: sharing (original) common API and common tests/examples
    - MySQL extensions within the MySQL package
- **Define CONDDB/POOL software component responsibilities**
  - POOL relational backend?
  - No direct dependency between POOL and CONDDB
    - But develop examples integrating the two *(which compiler? MySQL only?…)*
- **Define manpower and workload responsibilities**

Work plan items (2)

- **Design/circulate/prototype new common API**
  - Having already agreed what if anything should go to POOL
- **Oracle implementation issues**
  - Reengineer and speed up data insertion and retrieval
  - Must take the decision whether to keep or drop OCCI
- **MySQL implementation issues**
  - Should continue to support the many users
  - Keep in mind possible need for schema evolution or data migration…
- **Tools**
  - Export/import of data between Oracle and MySQL
  - Data browsing

THANK YOU

FOR YOUR

PRESENTATIONS

AND FEEDBACK!