Conditions & Configuration Database — a user point of view

Yan Benhamou, Erez Etzion, Daniel Lellouch,
Lorne Levinson, Shlomit Tarem

TGC Group

Conditions Database Workshop

December 2003

Summary of our point of view

- Scope of using the Conditions Database needs to be expanded. (It's not just for reconstruction.)
  - Leads to additional significant requirements
    - Principally we believe we need a full-function database with queries, keyed tables, reports
    - not just an Interval-of-Validity aware storage system for objects whose internal structure is inaccessible.
  - Argues for focus on flexibility,
    exact definition of 'Conditions' will evolve
- From the data content point-of-view the separation of Configuration and Conditions data bases is artificial and effort will be wasted in overcoming it.
  - Config data structures are stored/retrieved in CondDB and CondDB queries depend on them

Scope of use for the CondDB

- Supply the needs of the reconstruction program

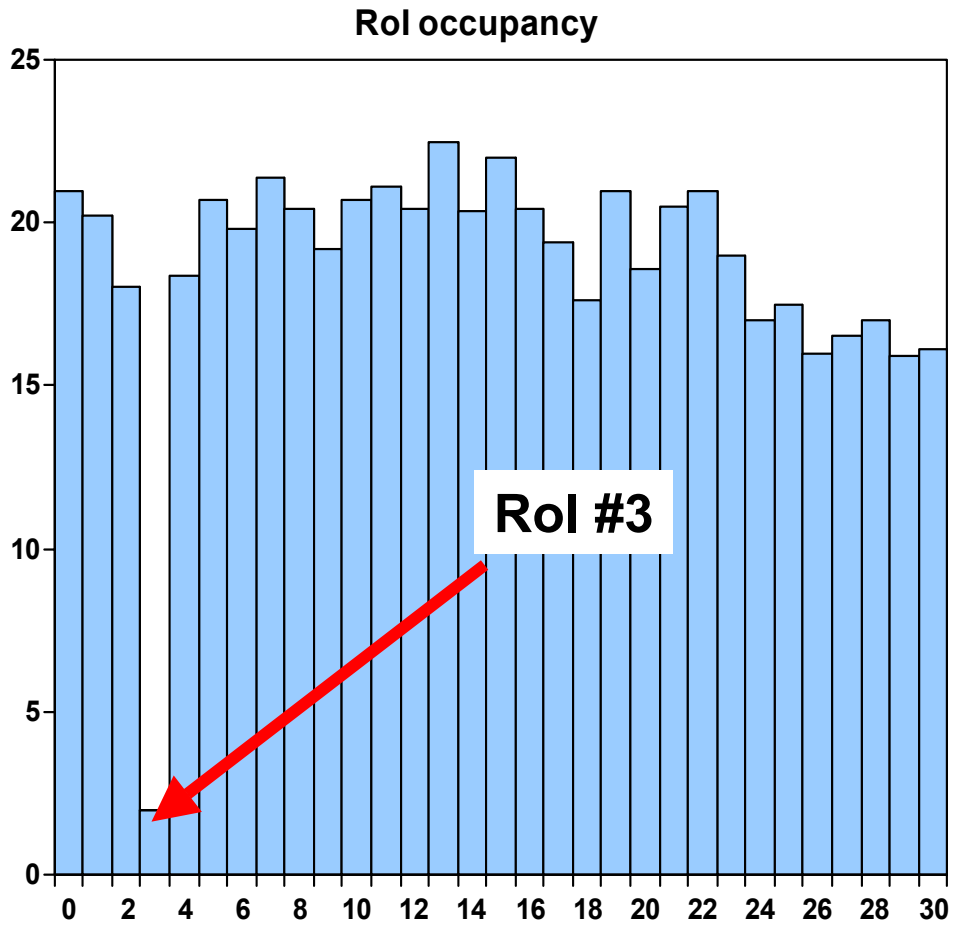
  BUT ALSO

- Information needed to diagnose detector problems,
  e.g. temperatures, voltages, TGC chamber charge
- Data that is needed as input to computations of calibration and alignment constants needed by reconstruction,
  e.g. the raw readings of alignment sensors
- Data for future online configuration constants,
  e.g. front end ASIC parameters, thresholds
- Input to detailed trigger simulation
  e.g. inefficient areas, hot or dead channels, programmable parameters of the trigger logic

An example:

Symptom of a problem seen in the Muon RoI histogram produced from the datastream, by a monitoring program:

Missing triggers in RoI 3



Expected user reaction:

- Based on all relevant factors (HV, thresholds, dead FE chip, wrong delay in FE chip, bad coincidence matrix,…) list all the potential objects that can result in such an effect, e.g.:
  - HV of chambers 523, 654 and 945 influence ROI 3
  - thresholds of …
- Retrieve the values of these specific items
- Locate the reason for the fault
- Look forward and back in time to determine the time interval of the fault
- Mark other affected quantities (one RoI may correspond to one defective HV which can in turn affect other RoIs)
  - Important for offline reprocessing
  - Absolutely vital for detector/trigger simulation (cross-sections, CP measurements, …)

BLOB vs SQL access

- Query the Configuration DB to know which items can potentially affect the identified problematic RoI.
- Dump all the blobs, parse them, and extract the values of the relevant items.
- How to do it?
  - Manually: very painful, though possible
  - Or, with a dedicated piece of code for every new problem encountered: Needs expert and might be a sink of computer resources if badly programmed
- The same is true for the process for marking all dependencies

One SQL command:

```
select thresholds
   from condDB
   where
   unit=(select unit
   from integration
   where ROI=3)
```

SQL query

- The ability to make such a query follows from the fact that:
  - each element has an entry
  - the hierarchy (chamber, DAQ and DCS, …) membership of the different elements can be described in the database by keyed tables
- The eventual expert system for system diagnoses will need to make such DB queries.
  - In fact it will construct such queries on-the-fly

Example II

- Due to a problem in a section of the detector, the section is taken offline for re-calibration of parameter X
  - During the re-calibration the configuration is changed
  - At the end of the calibration it is decided to ignore the re-calibration
- Meanwhile, it is found that the real reason for the problem was that the cavern background has become very high and one has to reduce the HV of chambers near the beam by 50V.
- So now we want to reload a configuration with:
  - Latest configuration including the new HV values
  - The values of parameter X prior to the re-calibration

Conclusions from Example II

- Old configurations stored in the Conditions DB are needed as the basis for current configurations
- Configurations have internal structure and components need to be extracted from various instances and combined into new instances

Furthermore:

- There will exist several standard versions of various sets of configuration components for different running conditions and run types, e.g.:
  - High cavern background
  - High/low luminosity
  - Diagnostic and calibration runs
  - …

How many databases?

- **All needed detector description data should have a single source.**

- Instead of saving labor, multiple APIs to multiple DBs in ATLAS could require **more** subdetector manpower to build and maintain:
  - Conditions DB
  - Configuration DB
  - Diagnostics and calibration DB

  ➔ Coherence of overlapping data
  - Software to synchronize the databases
    - How to trigger synchronization?
  - Import/export software

Instead

- Unify all these into one, probably partitioned, DB.
  - Most applications would use a small fraction of available information, with efficient DB performance
    - e.g. online instance: sub-detector specific, selected tables, short history
  - Adding new tables is easy and inexpensive, unless links (foreign keys) are to be followed in a query, but DB technology is far superior to user BLOB parsing.
- DB experts: please help us build our schemas.
- The API's should be optional toolkits, which do not exclude additional tables and direct SQL queries.
  - Example tool kit: Interval-of-Validity table schema
  - New Lisbon API 'tables' look promising,
    eagerly awaiting details, discussion how to apply to our needs