Database Strategy

Murrough Landon – 3 March 2005

`http://www.hep.ph.qmul.ac.uk/ landon/talks`

Overview

ATLAS database strategy

Existing L1Calo databases

Future plans and proposals

ATLAS database strategy (1)

Yes, there now is one!

database project: Richard Hawkings and Torre Wenaus

website: `cern.ch/Atlas/GROUPS/DATABASE/project`

strategy for online databases: ATLASDB-2004-7

`cern.ch/Atlas/GROUPS/DATABASE/project/online/doc/online_evolution.pdf`

ideas for subdetector configuration: ATLASDB-2004-10

`cern.ch/Atlas/GROUPS/DATABASE/project/online/doc/subdet_config.pdf`

occasional online database phone meetings

new TDAQ configuration WG: Igor Soloviev and Johannes Haller

recent workshop for database developers:

`agenda.cern.ch/fullAgenda.php?a044825`

ATLAS database strategy (2)

Outline vision

- general move to using common LHC Computing Grid (LCG) tools
- OKS configuration databases just for configuring TDAQ
- subdetector configuration from relational databases (if they want)
- new LCG conditions database project to replace “Lisbon API”
- TDAQ to provide access to these from within ROD crate DAQ etc
- there needs to be a compact subset of LCG libraries for online use

ATLAS database strategy (3)

Overview of LCG Tools

lots of layers with interfaces and plugin components

POOL for persistent objects (via ROOT for event storage)

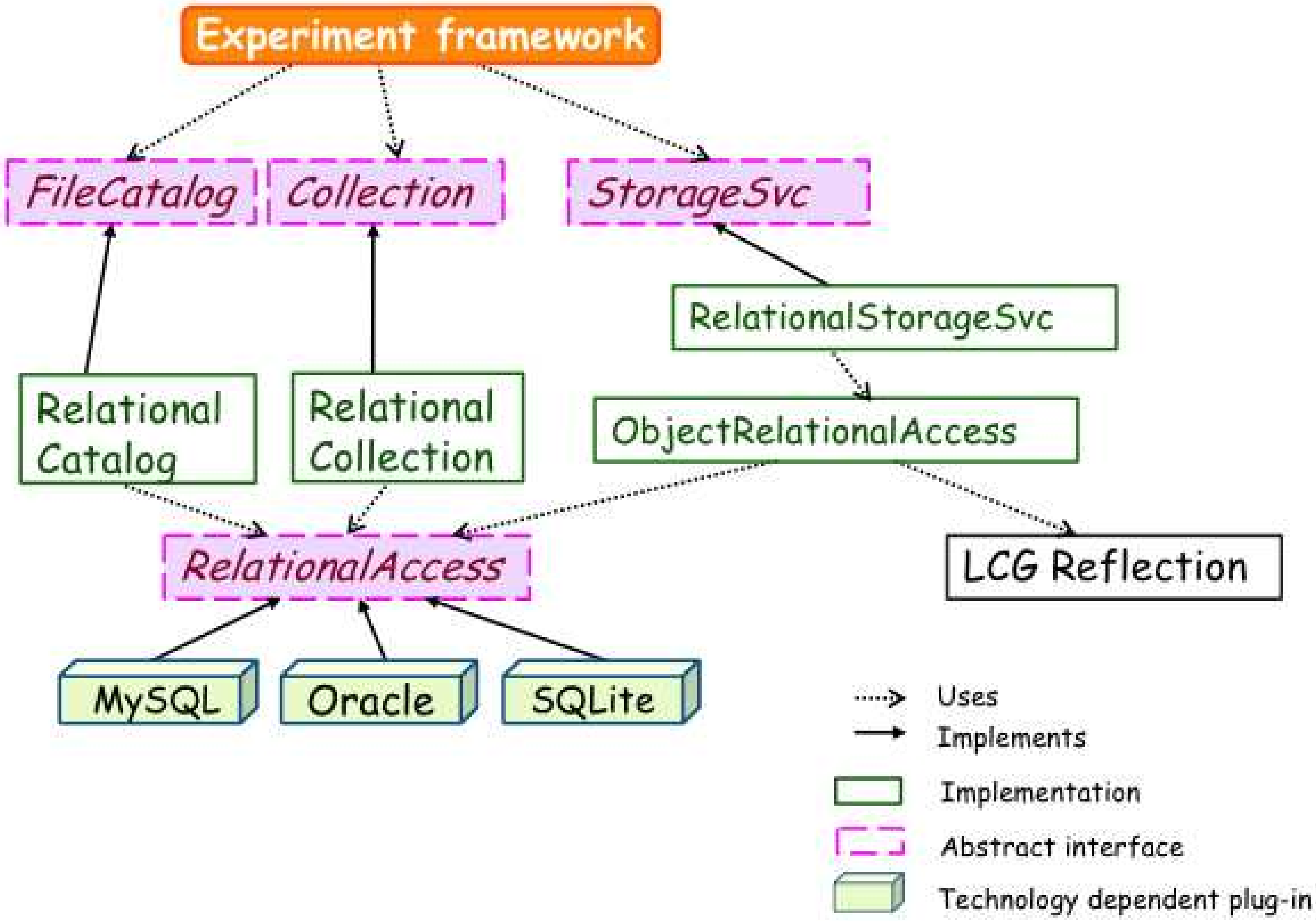
new “Relational Access Layer” (RAL): technology independent access to relational databases, implementations for Oracle, MySQL and SQLite

new Object Relational Access: store C++ objects using RAL

new conditions database (COOL): uses RAL plus interval of validity (per object)

hierarchical versioning system (HVS): give version to groups of objects

Overview of LCG Packages



ATLAS database strategy (4)

Relational Access Layer

abstract API for programmed access to RDBMS (largely avoiding SQL)

objects and methods for connecting to DB, starting/committing transactions, creating and dropping tables, inserting columns, adding and updating rows

released middle of 2004, extensively tested, some extensions requested

being used for new trigger menu developments

may be useful for configuration data that is easily mapped to relational tables by user code (but versioning still via conditions DB?)

ATLAS database strategy (5)

Object Relational Access

service for mapping C++ objects into relational tables

includes composite objects and STL containers (but not C arrays)

released late 2004

may be useful for configuration data that has a more complex object structure?
(but versioning still via conditions DB?)

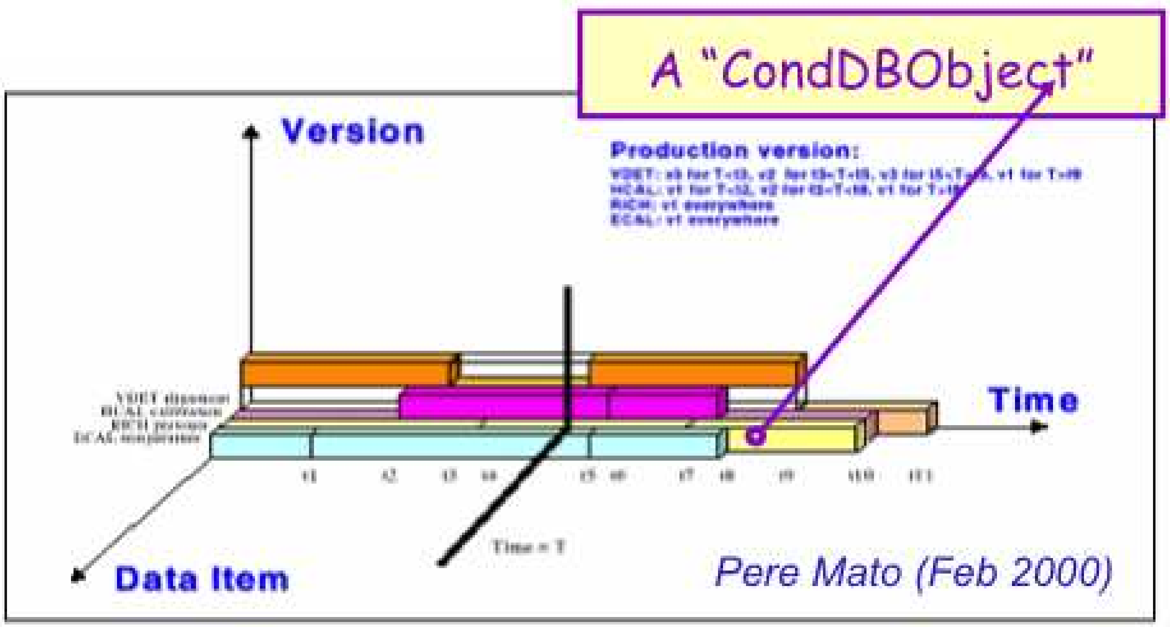
ATLAS database strategy (6)

LCG Conditions Database (COOL)

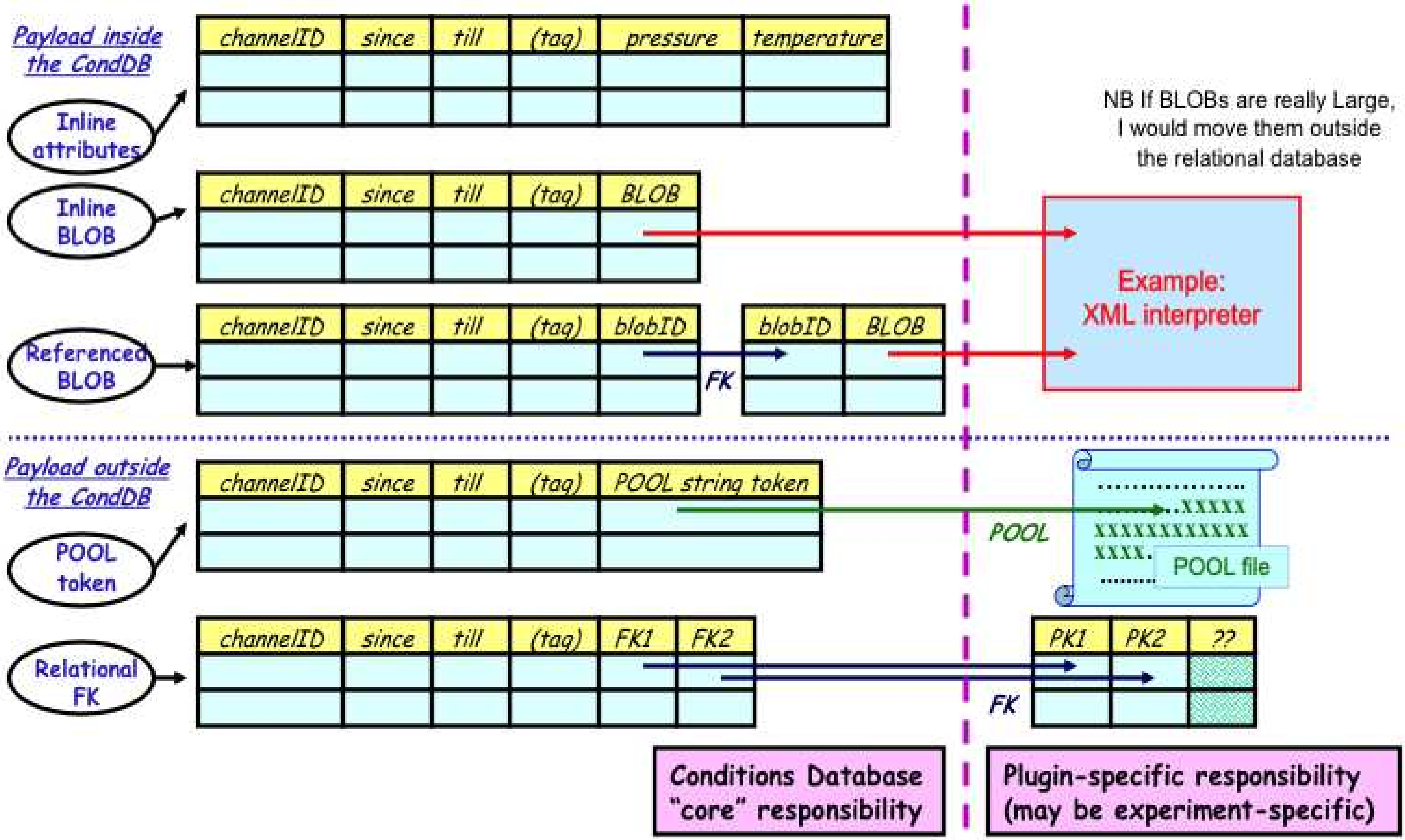
release expected by April 2005 (prerelease in January)

stores data values, objects(?), tables, BLOBs or references to data in other databases along with their intervals of validity

items organised in folders eg l1calo/calibration/physics/energy/ppm/nn



Possibilities for storing data in CondDb



Online/TDAQ database strategy

Future of configuration database?

present plan to keep separate configuration database for TDAQ

tdaq-01-01-00 removes possibility of configuring DB from process environment

perhaps move away from clients reading files to using RDB server

then RDB server could use relational DB as its backend instead of OKS files

thinking about ways to store state of configuration DB in conditions DB eg at start of run

L1Calo database (1)

Overview

so far everything is based on the online OKS configuration database

except for PPM configuration from separate XML structures

extensions for custom module types

description of firmware

cables

calibrations and trigger menu

run types

test vectors (DataGenRecipes)

some attributes taken from IS instead of static database

L1Calo database (2)

Partitions, segments, run control etc

fairly standard – additions include custom L1CaloSegment with run types and DataGenRecipes

likely to stay in OKS (or its evolution)

Software repository

description of programs and applications that use them

likely to stay in OKS (or its evolution)

however for us the SW repository also includes firmware so there is a link to hardware description (modules)

L1Calo database (3)

Hardware

detector, racks, crates, modules, workstations, cables

description of all hardware in USA15 will need to be in TC database

seems sensible to automatically derive hardware configuration from TC database (or upload the other way?)

but what about implications on firmware in modules, applications which run on particular hardware (crate CPUs etc)

Basic Use Cases

Online

read existing DB and load up to 20 crates simultaneously

edit/update DB, “by hand” (GUI tools) or by program

store used configuration in conditions DB

save new calibration in conditions DB

Offline

save/read calibration to/from conditions DB

read configuration and other conditions used online(?)

L1Calo plans (1)

Trigger menu

Johannes is implementing trigger menu for the whole of level 1 (HLT later) using RAL probably also with HVS for versioning

schema and classes inspired by our existing ones

should not be too hard to migrate when the new trigger menu DB is released – assuming support from RAL from ROD crate DAQ

L1Calo plans (2)

Calibration

we need to move from existing OKS calibration files to COOL

start asap after full release of COOL with access from next TDAQ release (April?)

need advice (discussion with Richard Hawkings?) about best way to organise our various types of calibration data in COOL

check how much of existing CMM, CPM, JEM calibrations are needed

what aspects of current PPM XML file are calibration as opposed to configuration choices? How do we handle the latter?

L1Calo plans (3)

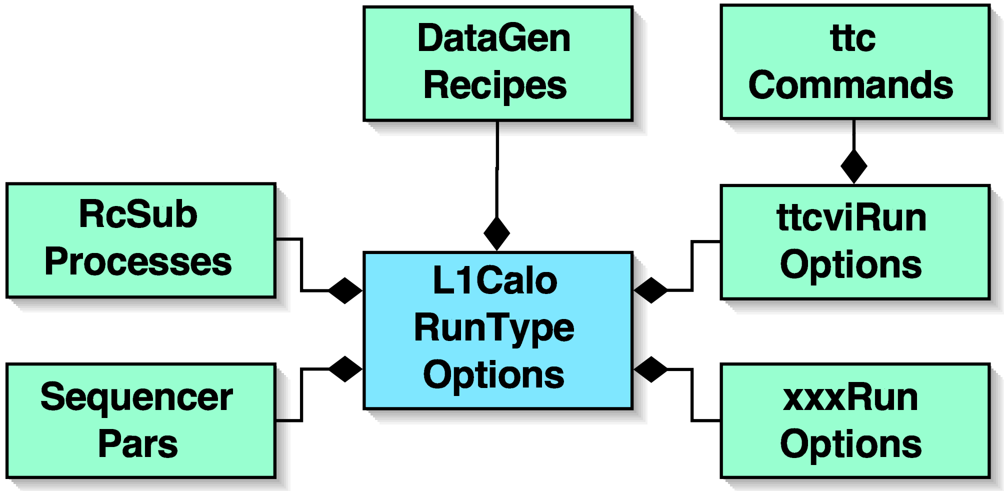
Run types

present schema allows of a lot of possibilities (mostly unused) but can be hard to edit

consider a simpler scheme? What do we really need?

also consider something allowing integration with similar concepts (if any) in the LAr/Tile to specify the parameters for joint calibrations

integrate run type and specification of sequences (whether in single run or multistep runs) and remove sequencer pars from IGUI?



L1Calo plans (4)

Hardware description

should come from TC database eventually – this will know about racks, crates, modules, cables

especially bulk description of cables – unless for final ATLAS installation the software should move to a purely algorithmic understanding of the expected connectivity?

lower priority than the rest?

L1Calo plans (5)

Other conditions

lists of dead/hot channels - so far we have little on this (apart from CPM and JEM channel masks)

what else?

Store complete configuration at run start?

read all registers of all modules and save to conditions DB?

ie in addition to saving higher level references to calibrations, trigger menu, run types etc that we used to load the system

CTP and MuCTPI plan to do this

rather more data for us

need a schema

L1Calo plans (6)

Use of IS

we have several overall parameters and a few for each module kept dynamically in IS

from last year the online model allows IGUI panels to write to the database and for run controllers to get updates

we didnt implement this – should we move towards that style now?

it would mean the (installed) database files on disks would reflect the most recent choices

...but not the original versions if kept as now in CVS

the database software would have a single source of information

could still retain a few purely IS parameters (run type, trigger menu etc?)

L1Calo plans (7)

API

at present all module services and simulation gets the database information for each module via a DbXXX object

methods returning either simple information or other objects

is this still the right model?

in future DbXXX could presumably return RAL or COOL objects

everything would depend on even more libraries...

...unless we use intermediate steps, eg RAL/COOL to private structures (XML or otherwise)

Calibration/configuration data (1)

CMM

calibration per module: TTCrx settings, readout pointers

calibration per input (BP/cable,pipeline): delays

configuration per module: geoaddress/firmware, num slices

configuration per input/output: mask (automatic at present)

CPM

calibration per module: TTCrx settings, readout pointers

calibration per channel: serialiser phase (unused?), serialiser delay (used?), cpchip phase (unused?), cpchip delay (unused?)

configuration per module: num slices, bcmux mode (via run type)

configuration per channel: mask per LVDS cable (automatic), FIO mask (automatic)

Calibration/configuration data (2)

JEM

calibration per module: TTCrx settings, readout pointers

calibration per channel: input phase, input delay, input mask

configuration per module: num slices

configuration per channel: FIO mask (automatic)

PPM

calibration per module: TTCrx settings, readout pointers

calibration per channel: PHOS4, DACs, FIR, LUT

configuration per module: num slices, bcmux mode(?)

configuration per channel: BCID options, mask, LVDS options

Calibration/configuration data (3)

ROD

configuration per module: busy threshold, max num RoIs

configuration per channel: firmware, mask