LCG Conditions Database Project

COOL Development and Deployment: Status and Plans

Andrea Valassi

On behalf of the COOL team

(A.V., D.Front, M.Clemencic, S.A.Schmidt, U.Moosbrugger)
Outline

• What are conditions data?

• Background to COOL

• COOL software overview and status

• Development and deployment perspectives
What are conditions data?

• Non-event detector data that **vary with time**
  
  – And may also exist in **different versions**

• **Data producers from both online and offline**
  
  – Geometry, readout, slow control, alignment, calibration...

• **Data used for event processing and more**
  
  – Detector experts
  – Alignment and calibration
  – Event reconstruction and analysis
Prehistory

• **Background of common activities in 2000-2003**
  - Collaboration of IT-DB, LHCb, Atlas, COMPASS, Harp
    • Borrowing a few design ideas from the BaBar experience
  - C++ API definition and Objectivity implementation
  - Oracle implementation of original API (“BLOB” payload)
  - MySQL implementation of extended API (flexible payload)

• **LCG Conditions DB project launched in 2003**
  - Subproject of LCG Persistency Framework (POOL)
Two parallel activities (common project mandate)

- Integrate existing Oracle/MySQL packages into LCG AA
- Review old software and API to plan new developments

Project faced two main problems in this phase

- Lack of manpower for new developments
- Divergence of the two packages

Decision to develop new software just after CHEP

- Following public AA Meeting discussion in October
- Development of COOL started in November 2004
“Common” project scope

NOT the problems specific to one experiment or one data type (handled by each experiment)

**Software for time-varying and versioned data: a common component with a well-defined task**
(RDBMS implementation of technology-neutral API)

NOT the generic C++ access to relational data (handled by CORAL)

NOT the generic deployment of relational services and distribution of relational data (handled by 3D – at CERN by IT-PSS)
Conditions data: metadata model

- Designed to handle data “objects” that
  - Can be classified into independent data items
  - **VARY WITH TIME**
  - May have different versions (for given time and data item)

A CondDBObject has
- Metadata:
  - Data item identifier
  - Interval-of-validity [since, until]
  - Version information
- Data “payload”:
  - Actual data variables (temperatures, calibration parameters…)
  - Separate fields or encoded as a LOB

**Main use case: fetch data valid at given time and tag**
- Inverse lookup (from temperature to time or version) is not a priority

This 3-D metadata model is still valid!
COOL software overview

• **Merge best ideas of two previous packages**
  – *New API enhances both original and ‘extended’ APIs*

• **Single implementation for many relational backends**
  – *Encapsulated behind C++ API (no direct SQL user access)*
  – *Support for Oracle, MySQL and SQLite via CORAL*
    • *Attention to Oracle performance (bulk operations, bind variables…)*

• **Maximize integration with other LCG projects**
  – *Reuse infrastructure and software (SPI, SEAL, POOL…)*
  – *Mutually beneficial collaboration with CORAL team*
COOL relational implementation

- **Modeling of condition data “objects”**
  - **System-managed common “metadata”**
    - Data items: many tables (“folders”), each with many “channels”
    - Interval of validity - IOV: since, until
    - Versioning information with handling of interval overlaps
  - **User-defined schema for “data payload”**
    - Support for simple C++ types as CORAL “AttributeList”

<table>
<thead>
<tr>
<th>objectID</th>
<th>channelID</th>
<th>since</th>
<th>until</th>
<th>pressure</th>
<th>temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Metadata**
System-controlled
(versioning metadata not shown)

**Data payload**
User-defined schema
(different tables for different schemas)
Milestones

• **Nov 2004: start of COOL software development**
  – Brand new code, merge ideas of two pre-existing packages
  – Initially ~1.3 FTE for development (A.V. and S.A.S.)

• **Apr 2005: first COOL production release 1.0.0**
  – Support for Oracle and MySQL through POOL RAL
  – Basic insertion/retrieval (single/multi-version, single/bulk)

• **Oct 2005: Atlas use case performance validation**
  – One job every 5s, each retrieving 100 MB in 100k rows

• **Latest of many releases: COOL 1.2.8 (Jan 2006)**
  – SQLite (July), CLOB, PyCool, multi-channel bulk ops (Aug.), performance tests (Oct.), data copy (Nov.), CORAL (Jan.)
  – Team has grown to ~3.5 FTE (IT/LCG, Atlas and LHCB)
Future perspectives

- **Software consolidation and enhancements**
  - Major API and schema changes in 1.3.0 (~Apr 2006)
  - Implement new features following experiment priorities

- **Support real life deployment before LHC startup**
  - Continue to test and improve Tier0 software performance
  - Support Atlas and LHCb in setting up distributed services
  - Collaboration with 3D and IT-PSS service teams is crucial
Software enhancement plans

• Next on the list (COOL 1.3.0 and later)
  – AMD64 port and storage precision in API (int32 vs. int64)
  – Add table with “channel” metadata; schema evolution tools
  – Improve versioning: user tags (later: HVS hierarchical tags)
  – Later: add CORAL monitoring/authentication/indirection

• More requests for new functionalities are pending
  – Often received at weekly phone meetings
    • Handled according to experiment priorities and available manpower
  – Formal review later on when API is more stable
Atlas (offline) deployment

- **COOL fully integrated into Athena since mid-2005**
  - Small payload stored ‘inline’, complex payload as POOL refs
  - Development priorities: CORAL API, schema evolution, HVS

- **Most data still in Lisbon MySQL implementation**
  - Transition phase: complete migration to COOL by mid-2006

- **Deployment priorities**
  - Commissioning (now); simulation (Apr); reconstruction (Oct)
  - Static replication now (Oracle->SQLite)
  - Explore T0-T1 dynamic replication via Oracle streams in 3D
    - Interest in distributed Oracle access via Frontier too
LHCb deployment

- **Conditions DB (COOL) one of many databases**
  - COOL holds conditions data for reconstruction/analysis
  - Other data in PVSS, LFC, Bookkeeping, Configuration DBs

- **Deployment model (online and offline)**
  - Masters (r/w) at the pit and CERN T0
  - Replicas (r/o) at T1
  - Oracle replication via Oracle streams

- **Data challenge plans in 2006**
  - Alignment/calibration challenge in Oct (with all T1 sites)
Summary

• **COOL software is recent but of production quality**
  
  – Software developments started in fall 2004 after CHEP04
  
  – Manpower has increased from ~1.3 to ~3.5 FTE
  
  – Single implementation for Oracle, MySQL and SQLite

• **Tight integration with other LCG projects**
  
  – Mutually beneficial collaboration with CORAL project
  
  – Service integration with IT-PSS at CERN and 3D project

• **Focus moving from development to deployment**
  
  – Development of new functionalities is far from finished
  
  – Deployment is progressing fast in Atlas and LHCb
For more information

- **LCG Conditions Database Project Web page**
  
  [http://lcgapp.cern.ch/project/CondDB](http://lcgapp.cern.ch/project/CondDB)

- **Related talks and posters at this conference**
  
  - COOL performance and distribution tests (A. Valassi)
  - CORAL relational database access software (I. Papadopoulos)
  - POOL object persistency into relational databases (G. Govi)
  - Software for a variable Atlas detector description (V. Tsulaia)
  - LHCb conditions database framework (M. Clemencic)
  - Database access in Atlas computing model (S. Vaniachine)