

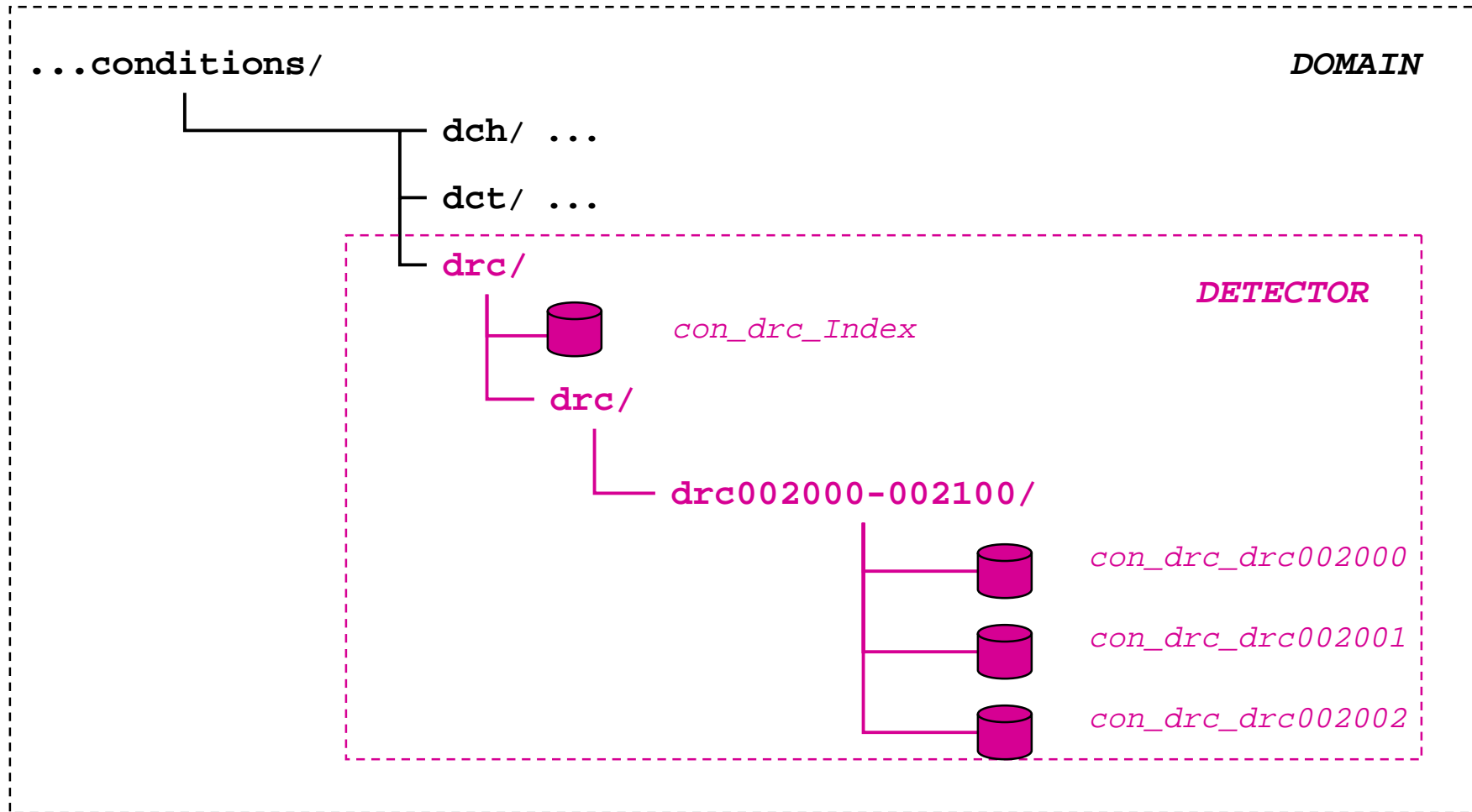


BABAR Software Week - 27th Apr 1999

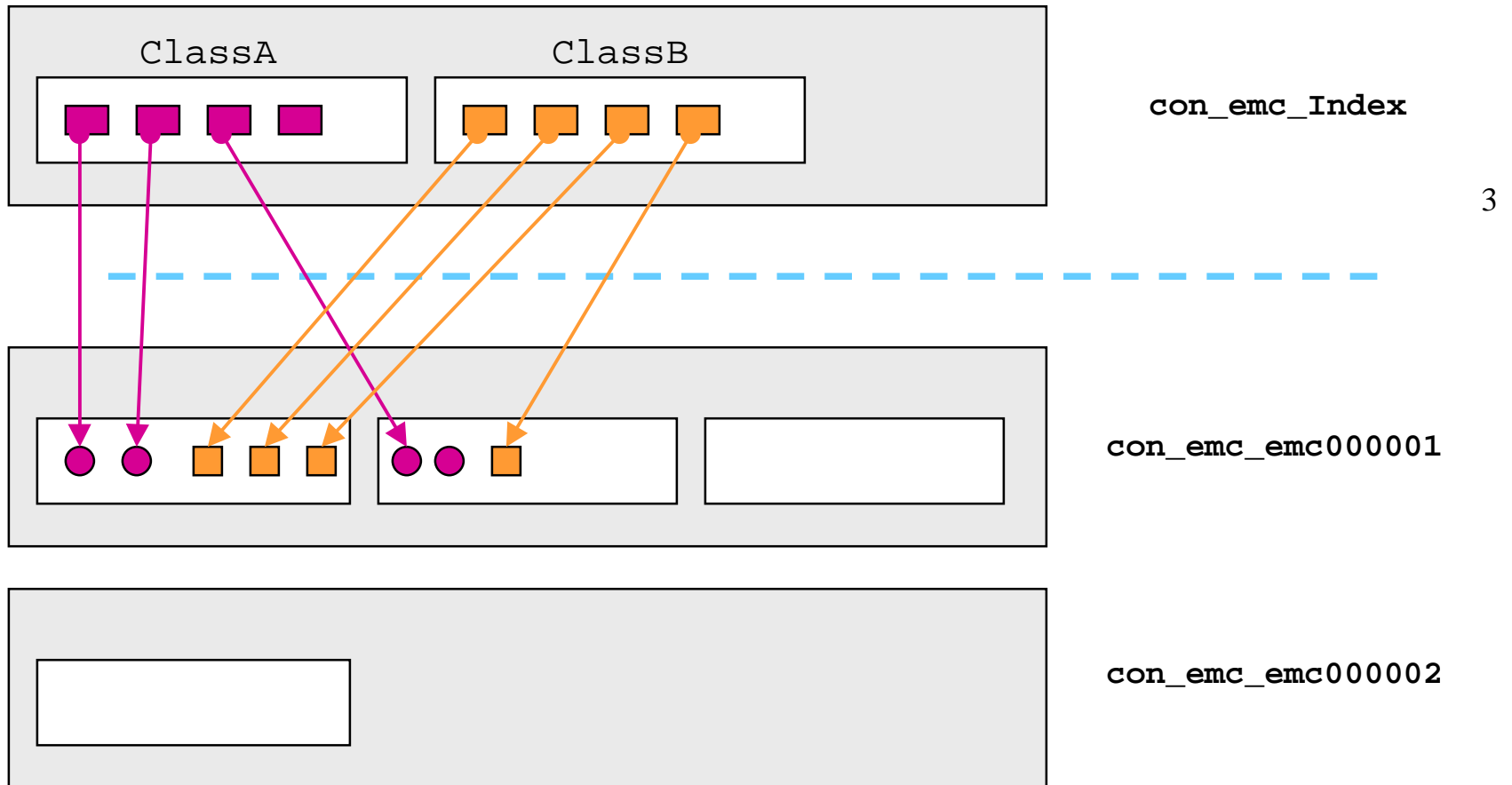
Data Distribution for Conditions *DB*

Igor A. Gaponenko
Lawrence Berkeley National Laboratory
(*IAGaponenko@LBL.Gov*)

Conditions DB structure



Logical links in Conditions DB



The Levels of Data Distribution

- **Level 1: Domain**
 - all the Conditions DBs from Federation A are transferred to Federation B
- **Level 2: Detector**
 - only a subset of Conditions DBs for a particular detector (“EMC”, “DCH”, ...) are transferred between federations
- **Level 3: Container (persistent class)**
 - only the information for specified persistent class (“EmcPulserCal”) is a subject for data exchange
- **Level 4: Revision**
 - only a subset (identified by “revision ID number”) of information for specified persistent class is a subject for data exchange

4

Level 1 (domain)

- **General Idea: => make a copy of all the database files**
- **Characteristics:**
 - no merging to existing DB on a remote site
 - all existing data (on remote site) will be destroyed/updated
 - this is unidirectional process
 - any (official) Conditions DB updates must be done at SLAC by running a loading application and then distributed following a standard schema

5

Level 2 (detector)

- **General Idea: => make a copy of a subset of database files for a particular detector**
- **Characteristics:**
 - no merging to existing DB on a remote site
 - all existing data (on remote site) for the corresponding detector will be destroyed/updated
 - this is unidirectional process
 - any (official) Conditions DB updates must be done at SLAC by running a loading application and then distributed

6

Level 3 (container)

- **General Idea: => extract all the information for a particular persistent class**
- The data are located through an interval container which keeps an update history for objects of a particular persistent class
- **Characteristics:**
 - possible merging to existing container on a remote site although the merging techniques does not seem a trivial exercise
 - existing data (written on remote site) for corresponding container could be left intact
 - could be bi-directional process

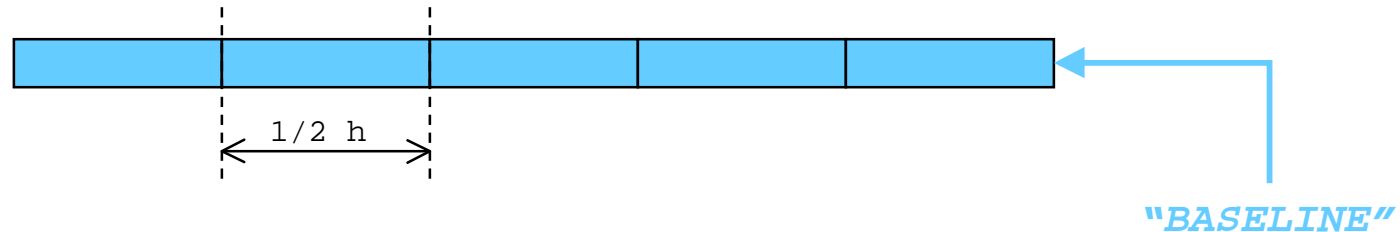
7

Level 4 (revision)

- **General Idea:** => extract a subset of information for a particular persistent class
- Only the data belonging to a particular “revision ID number” are selected for transfer
- This technique would allow to transfer the “increments” of conditions data between federations
- Characteristics:
 - easy merging to existing container on a remote site
 - existing data (written on remote site) for corresponding container could be left intact
 - is the easiest way to organize the bi-directional data exchange
 - revision IDs management across the collaboration will be required

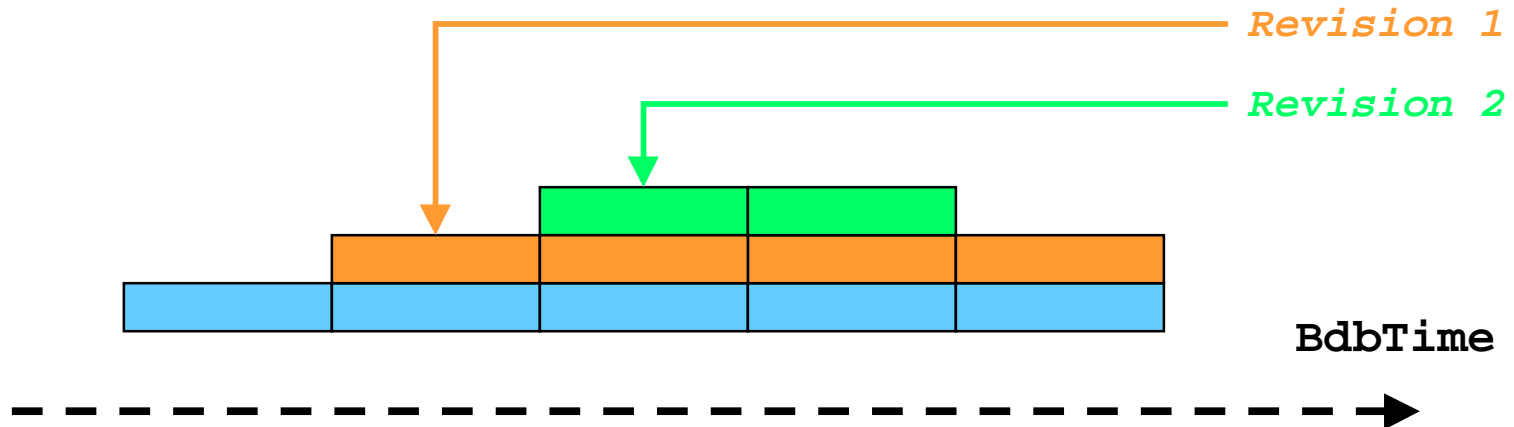
The idea of “revisions”

A “plain” history of a calibration object:



9

The history of a calibration object after 2 “revisions”:



Status

- There is a prototype for Levels 1 & 2
 - Exporting the domain/detector
 - ▶ BdbDmnExport
 - BdbDmnExport Conditions
 - BdbDmnExport -item emc:dch Conditions
 - Importing the domain/detector
 - ▶ BdbDmnImport
 - BdbDmnImport Conditions.export
 - The tools have passed some real tests (PRV0 -> SHIRE01)
- Working on other levels of data distribution
 - Level 3 (container) is expected within 2 weeks (no merging)
 - Level 4 (revision) will be 2 or 3 weeks after Level 3
 - ▶ Need to gain some experience with “Revision interface”. This is a new feature of the Conditions DB persistent schema and transient API

10

Open issues...

- The main technical and logical problem is merging data through interval containers
 - Would allow to modify and to keep intact locally created conditions data at a remote site. Do we need this?
 - Moving data back to SLAC. Do we need this?
 - Merging VS loading
 - ▶ load through bringing an application to SLAC and running it there against a central federation
 - Using “revisions” could help
- Potential problem of “distribution cycles” could be a management nightmare(?)
- Exchanging data between users’ federations(?)
- ???

11