



BABAR Collaboration Meeting - May 1997

Database Status Report

David R. Quarrie

Lawrence Berkeley National Laboratory

DRQuarrie@LBL.Gov



Overview

- Overall Design
- Prototypes
- Event API
- Data Distribution Strategy
- Data Logging Strategy
- Online databases
- Management Tools
- Schedule
- Personnel
- Summary



Overall Design

- Draft design documents on web
 - <http://www.slac.stanford.edu/BFROOT/doc/Computing/Databases/Roadmap.html>
 - Follow “Databases” link from *BABAR* Computing Page
- Integration of Event Store & Conditions Database
 - Different domains within *BABAR* Database Management System
 - Online databases is another domain
 - Domains are logically independent
 - Actually tied together by underlying Objectivity Federated Database
 - ▶ Also transaction boundaries



Authorization Levels

- Several authorization levels to protect database contents
 - System, Group, User
 - System level grants update access to complete database
 - Group level grants update access to particular group's databases
 - ▶ e.g. Subdetector (Conditions DB) or physics group (Event Store)
 - User level grants update access only to user's data
 - ▶ Not meaningful for Conditions DB
- Anyone has read access to complete database
 - Including all subdetectors, physics groups & other users data
- Authorization levels are additional level of protection
 - File access permissions also give protection



Event Collections

- Event collections
 - Small
 - ▶ Limited to $\sim 10^6$ events
 - Large
 - ▶ Limited to $\sim 10^{11}$ events (current implementation)
 - Run-based
 - ▶ Designed for possible data logging
 - Other implementations possible
 - ▶ Container-based, database-based etc.
 - All share common abstract interface (iteration etc.)
- Can be named



Event Dictionaries & Registry

- One dictionary per user, group and system
- Contain references to named event collections
 - Names must be unique within a dictionary
 - Names may have aliases (not yet implemented)
- Mechanism by which a collection may be located and used as a source of events
 - Application locates the desired dictionary & then the required event collection
- Event Registry
 - Set of all dictionaries & other management information

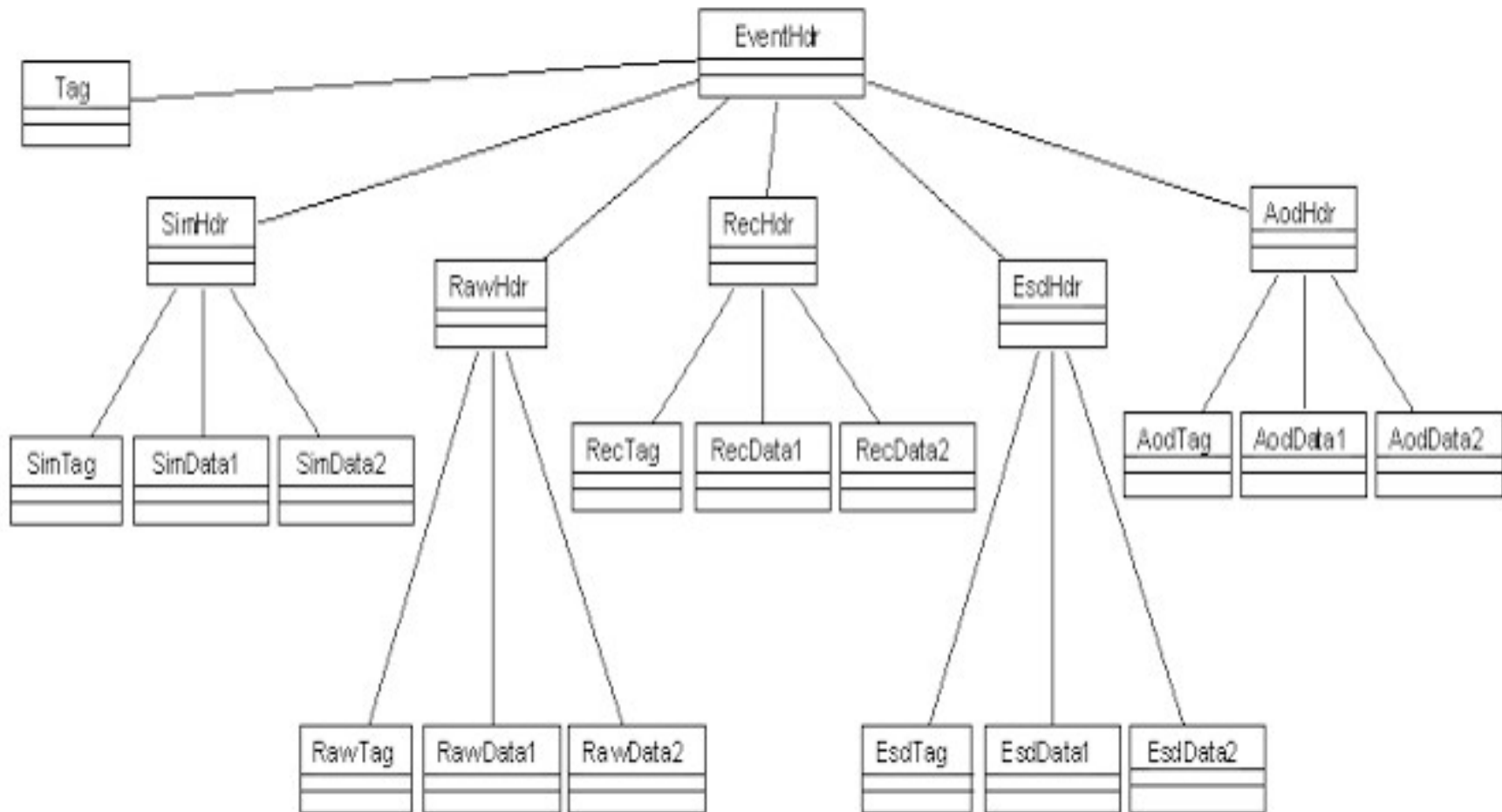


Events

- Hierarchy of objects - *event header, stage headers & tags*
 - Stages
 - ◆ SIM - Simulated Truth where applicable
 - ◆ RAW - Raw Data (~32KB)
 - ◆ REC - Reconstructed Data (~100KB)
 - ◆ ESD - Event Summary Data (~10KB)
 - ◆ AOD - Analysis Object Data (~1KB)
 - ◆ TAG - Event Tag (100-200 bytes)
 - Terminology taken from Atlas
 - Stage tags ~1-10% of size of corresponding stage information
 - Event Header (<64 bytes)
- Goal is to keep navigation information small



Event Hierarchy





Clustering Strategy

- Strategy for creation of objects
 - Objects likely to be accessed together should be located close together within the same database
 - ▶ Access of one object pre-fetches adjacent objects (page-based server)
- Strategy for creation of database files
 - Final database will have >50,000 files
 - ▶ Cannot create them all in a single Unix directory!
 - Spread them throughout a directory hierarchy
 - ▶ Basis for file protection based on authorization levels
 - ▶ Basis for caching/migration with HPSS



Status of Prototypes

- Several packages added to CVS with prefix *Bdb*
 - BdbApplication
 - ◆ Common application-classes
 - BdbEvent
 - ◆ Event Store classes
 - BdbCond
 - ◆ Conditions Database classes
 - BdbAccess
 - ◆ Test access programs
- RD45 package
 - Modified from CERN's original
 - ◆ DEC Unix added & several changes for our Makefiles



Status of Prototypes (2)

- Conditions DB
 - Test applications functional on AIX
 - ▶ Problems on DEC Unix being investigated
 - ▶ See report from David Brown
- Event Store
 - Test applications functional on AIX
 - ▶ Being built on DEC Unix now
- Authorization handling dummied out for now
- Clustering strategy not implemented
 - Limited database size until then



Emc Tests

- Using Emc Packages as testbed for Event API
 - EmcData, EmcReco, EmcEnv, *etc.*
- Database Event Input Module functional
 - Reads events from a designated event collection
 - Only functional with EmcGHit data for now
- Database Event Output Module functional
 - Loads event database from Dbio input file(s)
 - Only functional with EmcGHit data for now
- EmcMakeDigi modified to accept database-derived data
 - Testbed for Event API (see next slides)



Event API

- Exposure to database for reconstruction programmer?
- Trickiest decision so far
 - Transient
 - ◆ Hide persistence behind a transient API & copy data
 - ◆ No change to existing reconstruction modules
 - ◆ Restrictive on what can be part of the event
 - ◆ Rejected by Reconstruction Group
 - Persistent
 - ◆ Everything persistent-capable
 - ◆ Additional coding rules (d_Ref<T>, HepNew, HepDelete, *etc.*)
 - ◆ Most flexibility for adding data to event



Event API Testbed

- Emc packages are acting as a testbed
 - Understand all code changes
 - Understand migration issues
 - ▶ e.g. Changes to Dbio generated code
- Decide on what persistent classes we can use
 - RD45
 - ▶ HepEvent, HepVector, HepRefVector
 - Objectivity/RogueWave
 - ▶ Persistent versions of RogueWave classes
 - Based on RogueWave v6 instead of v7
 - Test yesterday showed could use at least some of them
- Decision end of May (feasible?)



Data Distribution Strategy

- Goals

- Efficient access to data from anywhere in collaboration
 - ◆ From bulk processing engines
 - ◆ From desktops in any Institution
- Flexible placement of data
 - ◆ Strategy should allow replication of data to optimize access

- Components

- Regional Centers
 - ◆ Secondary data repositories
 - ◆ Better bandwidth between Institutions & their Regional Center
- Database *Partitioning*
 - ◆ Replicate portions of database & act as local cache



Regional Centers

- SLAC
 - Main Collaboration Center (Regional Center for USA & Canada)
 - Primary repository for all data
- IN2P3
 - Primary European Regional Center
 - Replicates the complete data repository (present plan)
- RAL
 - Restricted subset of data
- INFN
 - Restricted subset of data



Database Partitions

- Replicate part of the *Federated Database*
 - Main partition
 - ▶ Master copy of the schema (C++ class definitions) & data
 - Secondary partitions
 - ▶ Copy of the schema & a subset of the data
- Secondary partitions act as a cache
 - When client accesses data, retrieve it from the “closest” location
 - ▶ Client & data in secondary \Rightarrow from secondary partition
 - ▶ Client in secondary & data not in secondary \Rightarrow from main partition
 - Transparent to client application
 - Database maintains synchronization between partitions

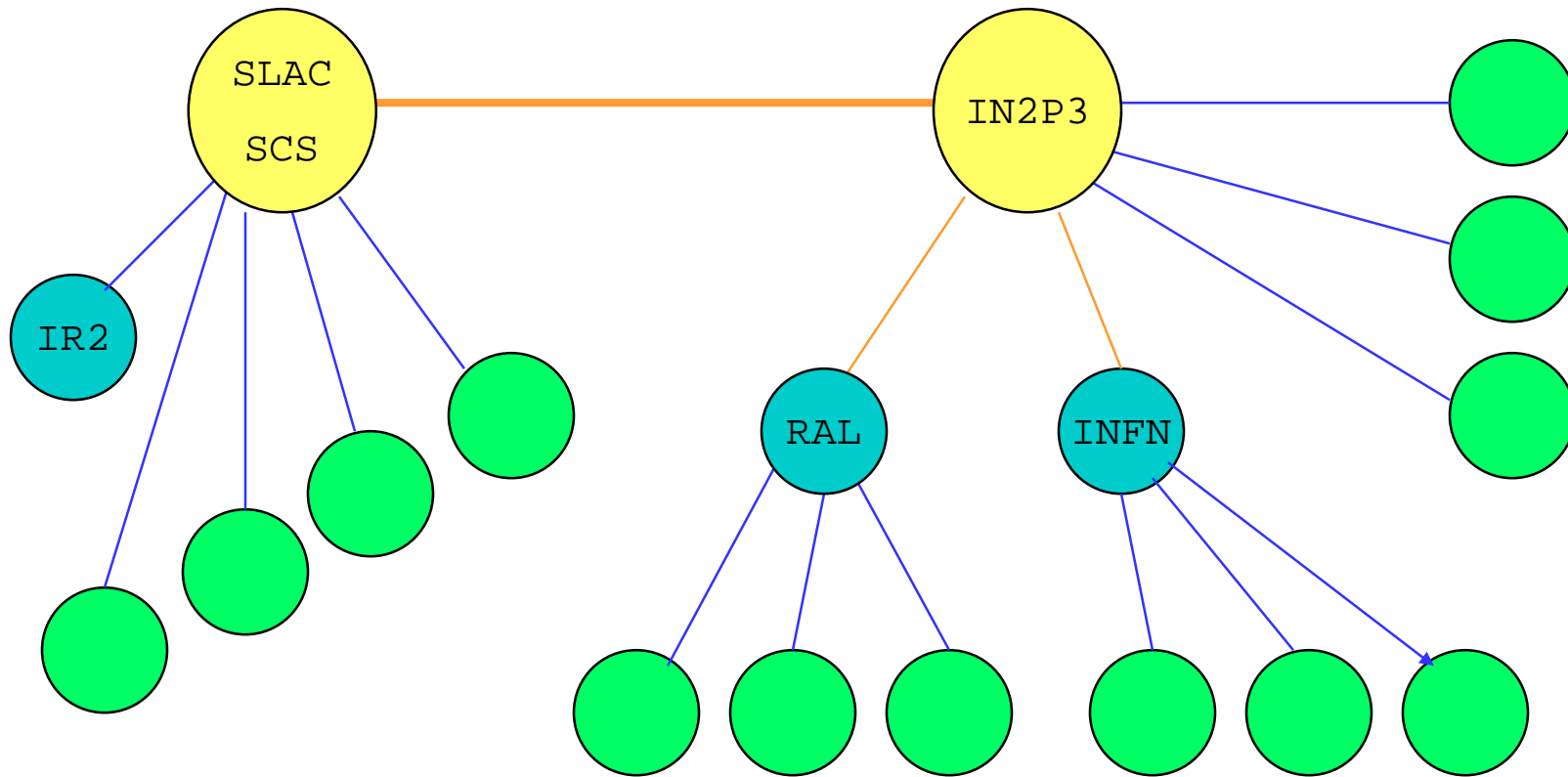


Database Partitions (2)

- Single Tier
 - Main partition can have multiple secondary partitions, but these themselves cannot have further children
- Database transactions are synchronized
 - Perform at the speed of the slowest secondary partition
 - Not a good idea to hold SLAC processing hostage to poor networking
- Ideal for IR2/SCS Synchronization of Conditions Database
 - IR2 as secondary partition with SCS as main partition



Data Distribution Strategy



— Tape transfers

— Partitioning



Data Distribution Strategy

- Strategy still evolving
 - Need experience with database partitioning
 - Issue of remote updating still unresolved
 - ▶ How to feedback to central repository data generated remotely
- Hope that Jean-Noel Albert from IN2P3 will be able to work on defining the strategy



Data Logging

- Current strategy (changing...)
 - OEP outputs streamed event data
 - ◆ *Not* objects
 - Spooled to conventional disk files
 - ◆ Location of spool disk?
 - ◆ Stage to tape if disk becomes full
 - Assume persistent Event API
 - ◆ Reconstruction requires persistent event objects
 - Separate database loading stage prior to PR?
 - Streamed data input module objectify the data & output it to database?
 - Prompt Reconstruction inputs from database & outputs to database
 - ◆ First contact with HPSS (or database loading stage)



Data Logging - Conditions DB

- Both online & offline require access to conditions database
- Proposal is to make online a secondary Objectivity partition
 - SCS is primary partition
- Updates from online automatically propagated to offline
- Updates from offline automatically propagated to online
- Semi-autonomous in case of network failure



Online Databases

- Another hole
- Chris Day & I think we understand how we could use Objectivity for online partitioning
- Several other possible roles within online
- Java interface?
- SQL Interface?
- Ray Cowan has expressed an interest in online databases



Management Tools

- Database management tools
 - Monitor usage & access efficiency
 - Cleanup database
 - Perform replication or re-clustering etc.
 - User Interface rather than programmatic interface
- Large Hole at the moment
 - Long term schedule details tasks but nothing available until next year
 - Hope can use SLAC Database/Web expertise to help us develop user interfaces



Schedule

- Preliminary schedule available
- Available from Databases Web pages
 - <http://www.slac.stanford.edu/BFROOT/doc/Computing/Databases/RoadMap.htm>
- Needs updating
 - Schedule up until June poorly met
 - Present progress indicates that schedule after June looks more realistic



Personnel

- Significant improvement since December
- Cast of characters
 - Pavel Binko, David Brown, Akbar Mokhtarani, Jim Ohnemus, David Quarrie, Alex Romosan, Todd Stavish (LBNL)
 - Paul Raines, Andy Hanushevsky (& more to come from SCS) (SLAC)
 - Stephen Gowdy (U. Edin.)
 - Ed Frank (U. Penn.)
- Not all 100%
- More to come
 - Ray Cowan (SLAC), Jean-Noel Albert (Orsay)



Summary

- Good progress on overall design
 - Integration of database domains
 - Testbed prototypes
- Event API has been a really tough decision
 - Recent progress gives hope for light at end of the tunnel
- Much still to be understood about data distribution
- Scaling & HPSS integration issues
 - RD45/Objy/HPSS Meeting at Objectivity tomorrow
 - Objectivity WorldView Conference at end of this week
- Manpower coming into place
- Schedule looks tight but doable