



*Reconstruction Workshop - 14th Nov 1997*

# *BABAR OO Databases*

---

David R. Quarrie

Lawrence Berkeley National Laboratory

*DRQuarrie@LBL.Gov*



# Overview

---

---

- Background
- Conditions Database
  - Review of Design
  - Current Status
  - What's next?
- Event Store
  - Review of Design
  - Current Status
  - What's next?
- Summary



# *Objectivity/DB*

---

---

- Technology that all the OO databases are based on
- Basis of RD45 collaboration from CERN
- Our approach has been to demonstrate that it will fulfill all roles for us, even if an alternative technology might have been as suitable for one of the several roles.
- Collaboration with RD45 can help but cannot be the complete answer
  - Very different timescales



# Terminology

---

---

- Persistent Object
- Page
  - Unit of transfer to/from data storage from client application
  - 8k to 64k bytes
- Container
  - Clustering/locking unit
  - Up to 32k pages
- Database
  - Maps (today) to a Unix file
  - In future Objectivity release Unix file will map to 1:n containers
  - Limitation of 65535 databases per federation (next slide)



# Terminology (2)

---

- Federation or Federated Database
  - Multiple databases distributed over multiple data servers
  - Disjoint federations cannot communicate directly
    - ▶ Can transfer files between federations (with some limitations)
- Schema
  - Persistent Classes
- Object Versioning
  - Ability to store multiple versions of objects in federation
    - ▶ c.f. CVS
- Schema evolution & versioning
  - Evolution: Change class structure & optionally migrate data
  - Versioning: Add new classes



# Overall Design

---

---

- BABAR DBMS covers several *domains*
  - Event Store
  - Conditions Database
  - Online Databases
- Objectivity is underlying technology
- Domains are logically independent
  - Actually tied together by underlying Objectivity Federated Database
    - ▶ Also transaction boundaries
- Allows independent development with late binding



# Conditions Database

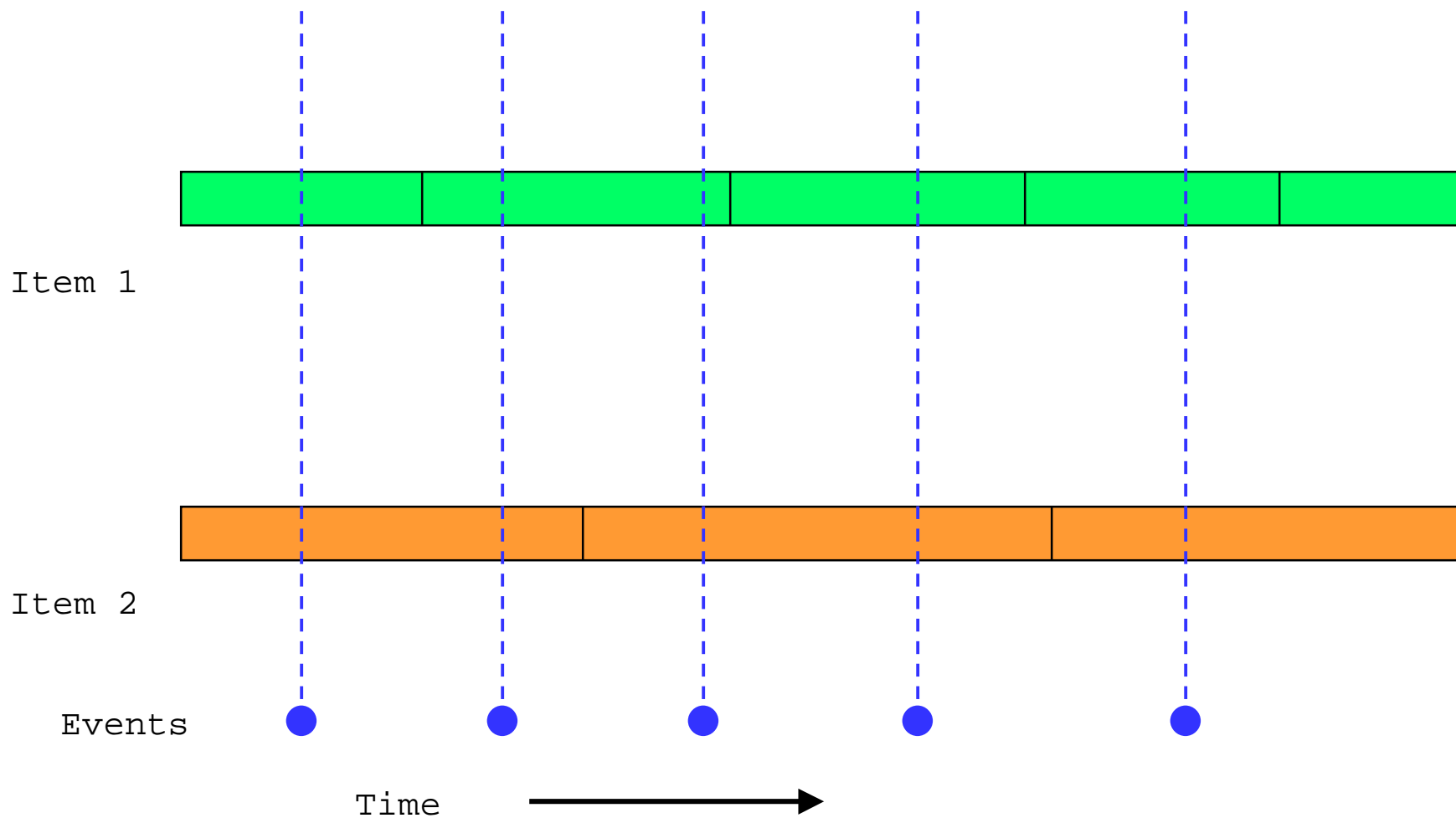
---

---

- Repository for history under which the data were acquired & processed
  - Electronics calibrations
  - Detector alignments
  - Trigger/Online/Detector configuration
  - Reconstruction adjustable parameters
- Time interval based
  - Items assumed stable for intervals with well defined start & end times
  - Each event tagged with time
- Time is primary key

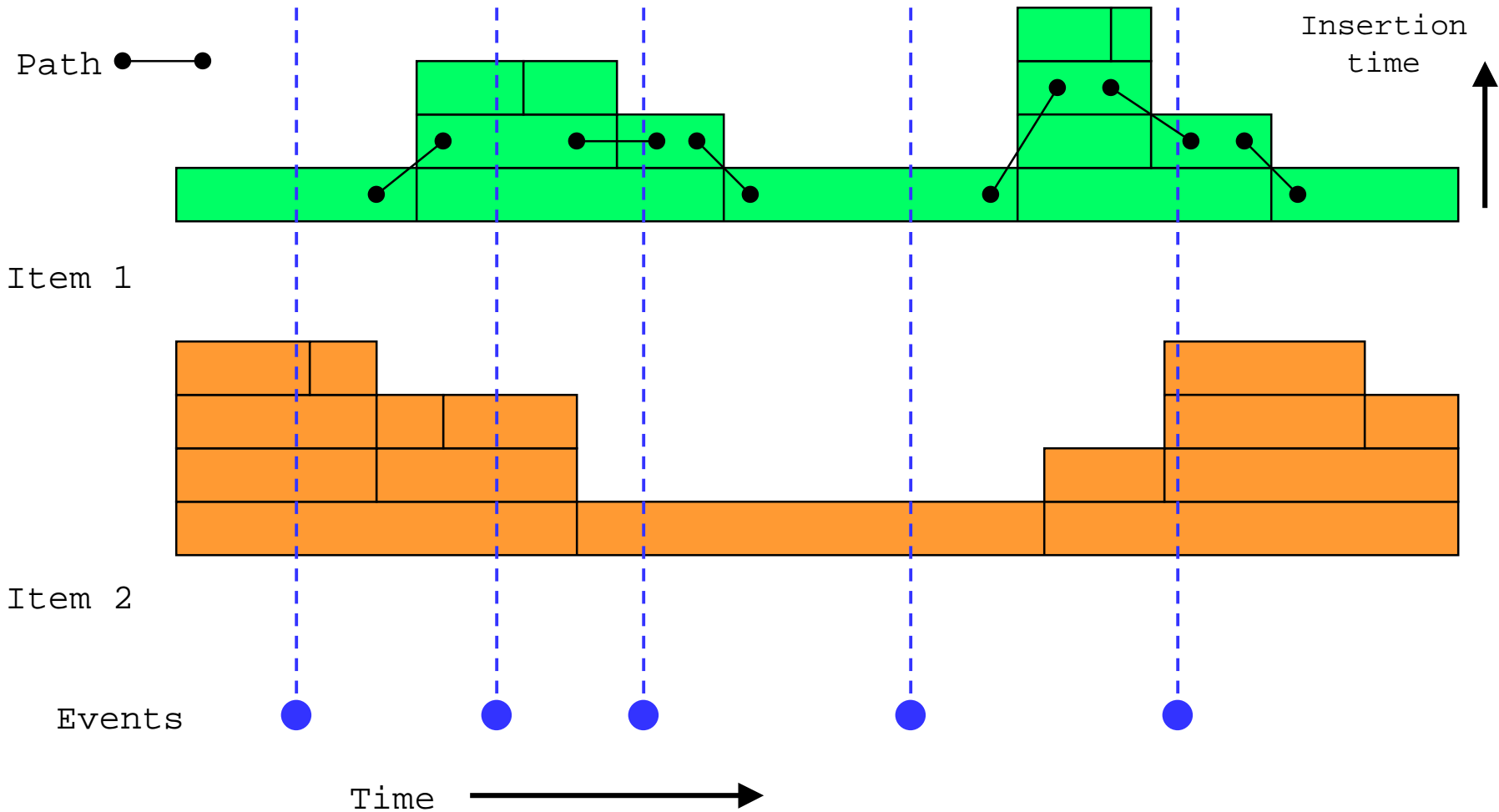


# Conditions Database Design





# Conditions Database Versioning





# Conditions Database Status

---

---

- Non-versioning release in Production Release since 4.4.1
  - 30 Aug 1997
- Versioning release going in now
  - Full API not yet available
    - ▶ Iteration across versions
  - Default is most recent version
- Still some open issues on how to specify a path
  - Current plan is to use 2nd time axis
    - ▶ Insertion Time
    - ▶ When new version created - new insertion start time
      - Overrides insertion end time for previous version
  - Specify path by specifying an insertion time



# *Event Store*

## *Collections, Dictionaries & Registry*

---

- Event collections “hold” event samples
  - Contain references to event headers
- 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



# Authorization Levels & Event Data

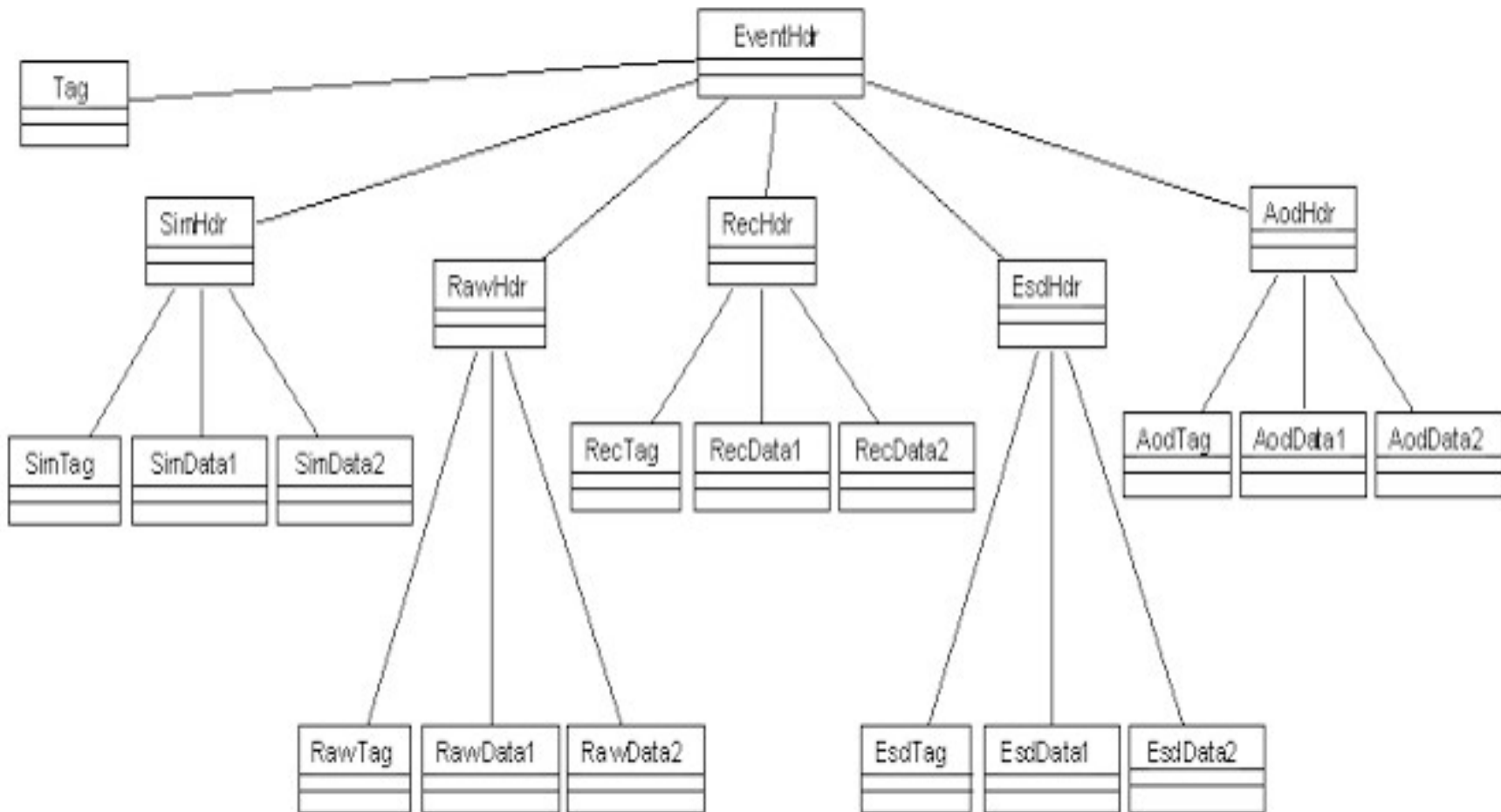
---

---

- Within Production Federation
- System level collections & event data
  - Collaboration wide
- Group level collections & event data
  - Physics Groups?
- Each user has ability to create their own collections & event data
  - Subject to space restrictions
- Constraint is that all must use production schema
  - See later slide



# Event Hierarchy





# Event Structure

---

---

- Compact
  - Minimize database overhead
- Hierarchical
  - Allow access to portions of the event with minimal overhead
    - ▶ Navigation nodes small
    - ▶ Leaf nodes in different databases
- Extensible
  - Allow us to add new data as our understanding improves with minimal impact



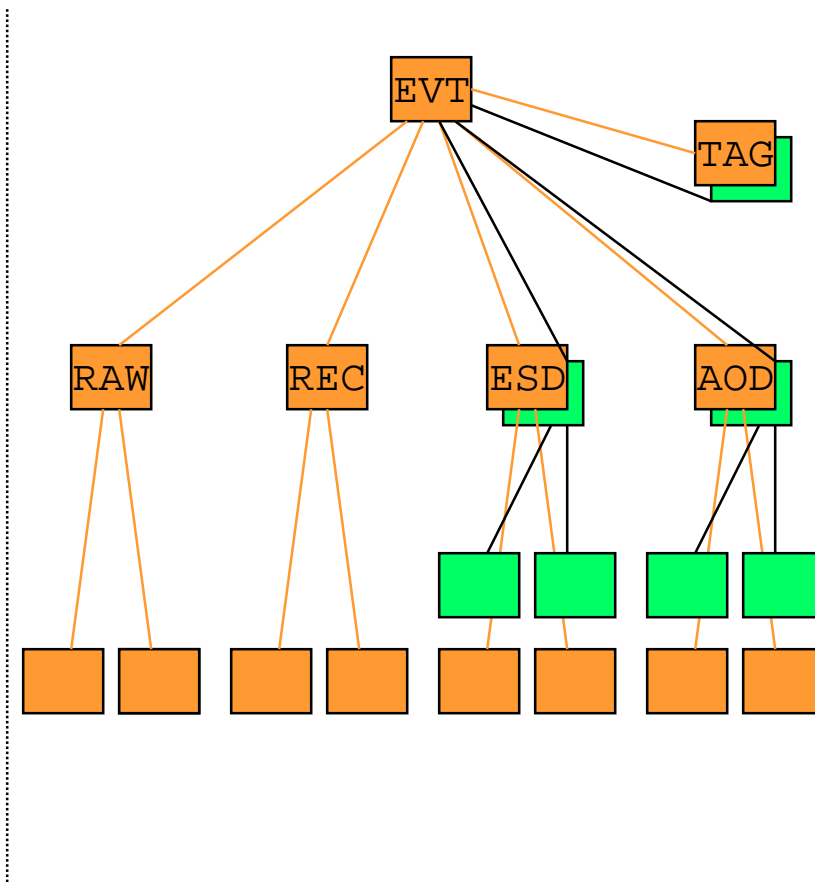
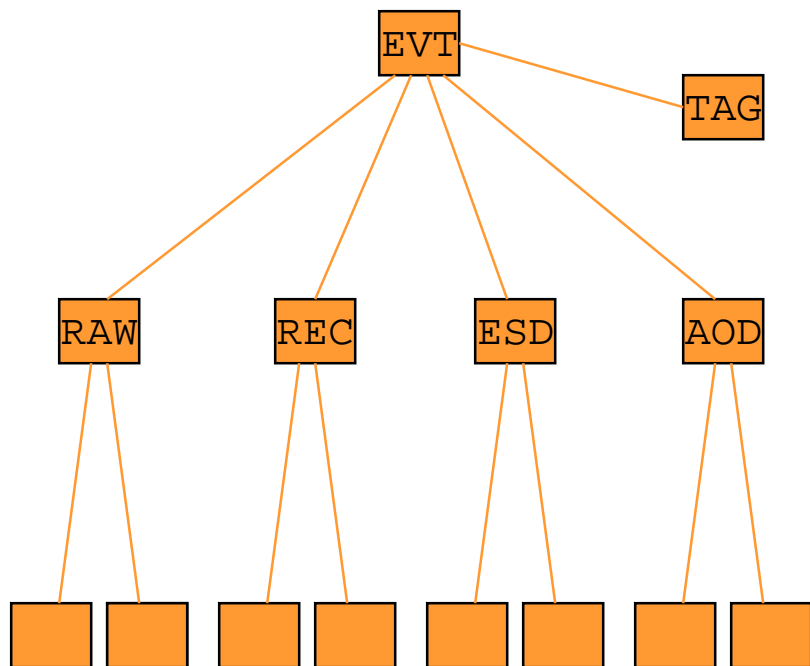
# *Event Reprocessing*

---

- Support for reprocessing
  - Not yet incorporated, but intend to use same technology as conditions database
    - ▶ Version Tree Headers
    - ▶ Create new information under new headers
- Effect of reprocessing depends on whether input collection is the same as the output collection
  - Input & Output the same
    - ▶ Tree Headers are versioned & new data hung off new versions
  - Input & Output are not the same
    - ▶ New Event Header & relevant Tree Headers created
    - ▶ New Event Header has references back to old Tree Headers as well

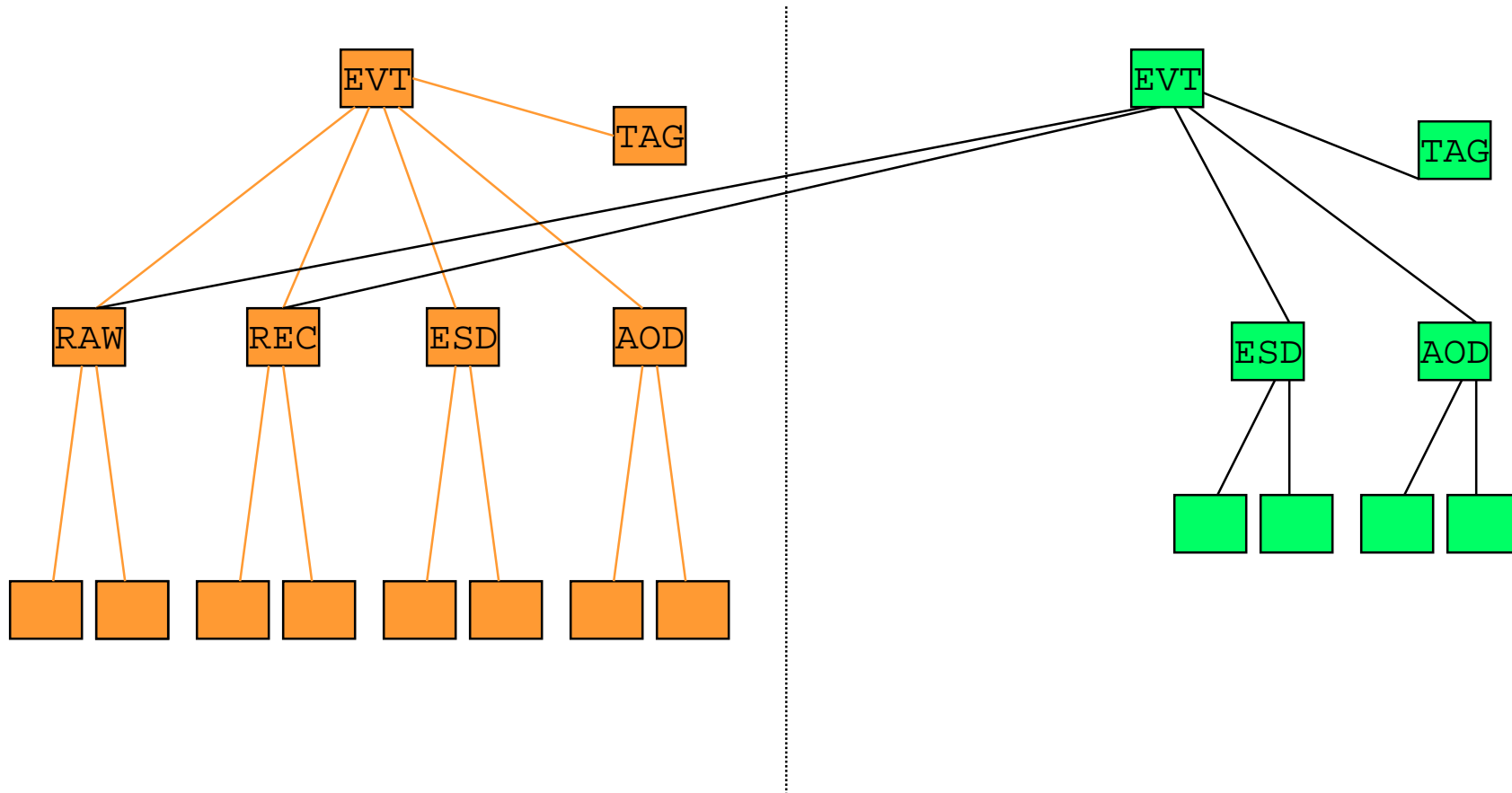


# Event Reprocessing - Same Collections





# Event Reprocessing - Different Collections





# *Event Store & Reconstruction*

---

---

- Start at simulated data & move towards physics analysis
- All GHit classes supported
- More of the Emc classes supported
  - Through EmcBump
- Some of the StdHep classes supported
- Tracking starting
- Other detectors need to ramp up quickly
- Not yet tested input to Beta
- Not yet much activity to support physics analysis
  - Simon Patton joining group in December will work on this



# *Reconstruction User Interface to Database*

---

---

- Database Input Module
  - User interface to specify an event collection name
- Database Output Module
  - User interface to specify an event collection name
- \*BdbSequence Modules
  - Enable/disable access to & generation of persistent data
- Everything else should remain the same



# *Event Store Development Status*

---

---

- Two phases
  - Development
    - ▶ Each new release is independent of previous ones
    - ▶ No data can be propagated forwards between releases
    - ▶ No controls over schema evolution
  - Production
    - ▶ Each new release is based upon schema & data of previous one
    - ▶ Data can be propagated forwards between releases
    - ▶ Schema evolution has to be managed
      - Not just changes to existing schema, also addition of new schema
  - Currently in development phase
    - ▶ Transition?
      - Before MDC II



# Event Store Release Status

---

- Release builds in parallel with regular releases
  - Currently AIX, OSF & Solaris
  - HP-UX waiting for Objectivity support of aCC compiler
    - ▶ Beta test this month
- Still some problems with dependencies
  - Tied in with SRT upgrades
- Thinking about making these -Objy releases the *only* releases
  - Half the workload of Release Coordinator
  - Needs more testing



# Data Distribution

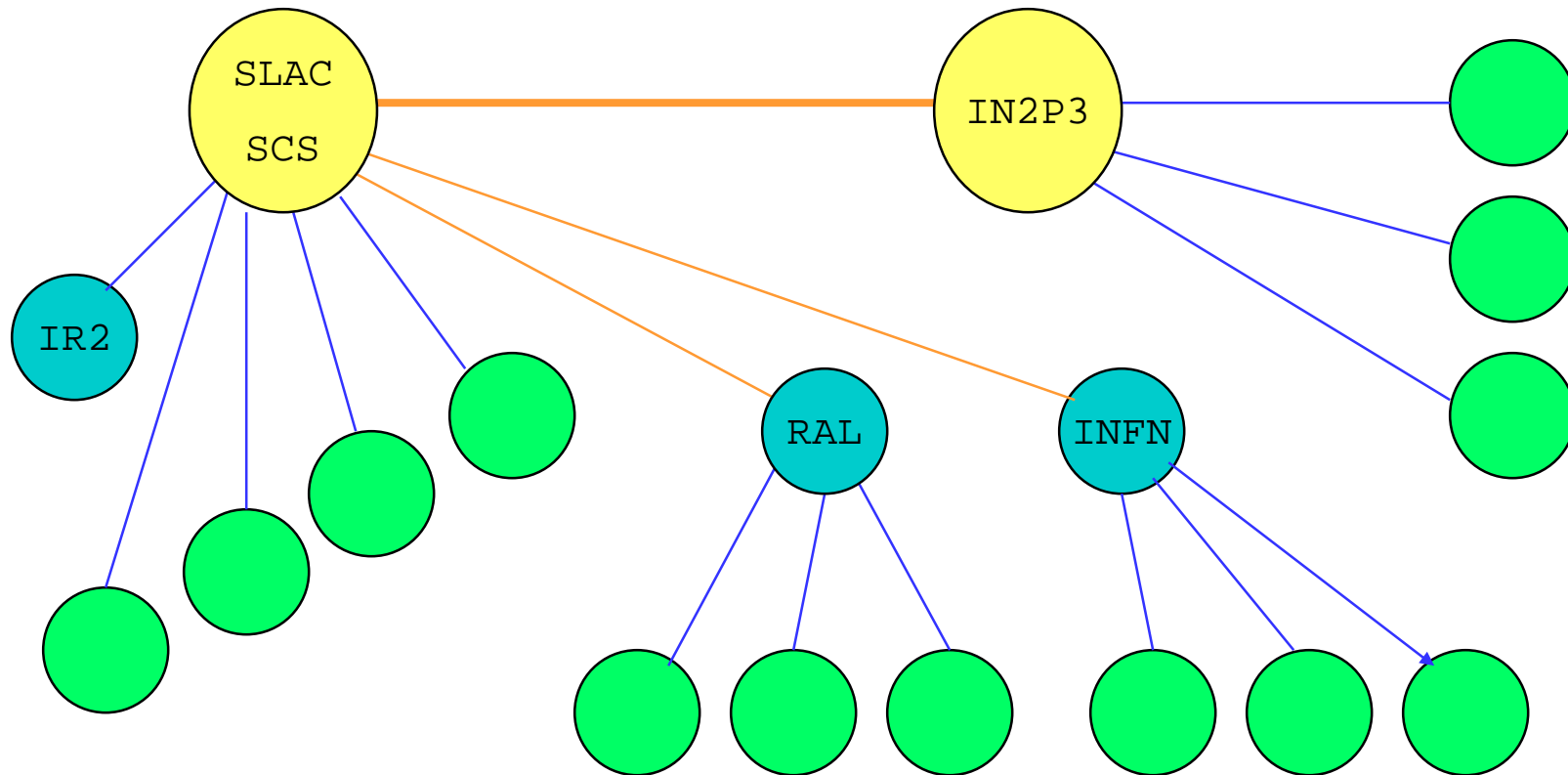
---

---

- Part of placement strategy
  - Place data close to where it is being accessed
- Issues of network reliability & bandwidth
- Role of Regional Centers
  - IN2P2, RAL, INFN
- Role of SLAC
  - Regional center for USA/Canada
    - ▶ I have some concern about interference with online logging & production



# Data Distribution Strategy



— Tape transfers

— Partitioning?



## *Data Distribution (2)*

---

---

- Bulk exporting/importing of production data
- Data sharing between people doing physics analysis
- Two different mechanisms being looked at
  - Preallocation of databases with reservation scheme
    - ◆ Suitable for bulk transfer of data to e.g. regional center
    - ◆ Inefficient for small transfers
      - Data for a single event scattered across many databases
    - ◆ Next release of Objy might make preallocation unnecessary!
  - Set of interchange databases with intelligent object copy
    - ◆ Suitable for physics analysis data exchange
    - ◆ Efficient for small transfers - inefficient for large



# Data Protection & Safety

---

---

- Need to protect data against accidental corruption or deletion
- Concept of authorization levels to protect database contents
  - Domain specific
    - ▶ System, Group, User
  - You have update access only to information for which you have been registered
- Everyone has read access to complete database
  - Including all subdetectors, physics groups & other users data
  - Don't (yet) exclude non-*BABAR* people
- Also adding file protections (chmod)



# *Federated Database Strategy*

---

- Protect master federated database
  - Contains schema and database catalog
- *Reference* federated database
  - Contains schema only
  - Used for builds of new *BABAR* software releases
  - Schema evolution disabled by default - managed evolution.
- *Production* federated database
  - Database catalog
- *Developer* federated databases
  - Copy per developer, copy of *Reference* FDDB
- Federated DB ID allocation scheme underway



# Schema Management

---

- Incorporating new classes into production
  - Allow validation checks & formal acceptance procedure
    - ▶ Management inertia
  - Accept no data into production federation that wasn't created with production schema.
  - Individual developers can add their own schema to their local federation & generate data using it.
- Sharing of schema & data between developers
  - Don't yet know how to share both data & schema between developers because of possibility of class Type ID clash if they have both added new classes
    - ▶ No problem if willing to regenerate the data



# Schema Management (2)

---

---

- How do we deal with modifications to our existing schema?
  - Schema evolution or class versioning?
  - Declare that it can't happen & create new classes?
  - Updating of existing data?
- Extensible event structure is designed to minimize schema evolution
  - Also breaks dependencies



# 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



# *Performance Monitoring Tools*

---

---

- Objectivity gives you some statistics gathering tools
- Not much thought gone into this yet
- I think it's crucial
- Part of the motivation for using BdbRef(T) rather than ooRef(T) *etc.* was to allow for some monitoring to be added
- Intend to spend more time on this in the next 6 months



# Summary

---

---

- Making progress
- Database builds in the Release procedure on a regular basis
- Still very much in a development phase
- Don't yet know how to do some things that we'd like to be able to to
- Have started at front end of food chain and need to move down towards physics analysis
  - RD45 have been working on some ideas for physics analysis