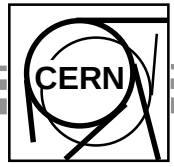
# *Building a multi-PB Object Database for the LHC*

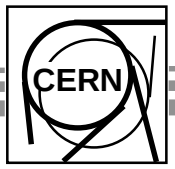
**The RD45 Project**

Jamie Shiers
Application Software and Databases Group
Information Technology Division
CERN



# *Overview*

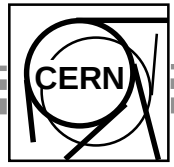
- The Large Hadron Collider (LHC)
- The RD45 Project
- Choices: technologies and standards
- Building a multi-PB object store
- Product choices
- Scalability and performance
- Interface to Mass Storage
- Short term plans
- Risk Factors
- Summary and Conclusions



# *The Large Hadron Collider*

- New facility currently under construction at CERN
- Scheduled to enter operation in ~2005
  - will operate for 15-20 years
- Will re-use existing infrastructure
  - including 27km long tunnel currently housing LEP
- Some 4 large experiments planned for the LHC
  - ATLAS, CMS, ALICE, LHC-B
- Will generate data volumes of ~5PB/year at data rates of 100MB/second - 1.5GB/second
- Gives rise to a total data volume of some **100PB**

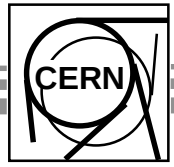
💣Timescale & Lifetime will be equally difficult problems



# *Data Volumes & Rates*

Data volumes and rates for 1 LHC experiment
Total volume per year ~5PB
Total LHC data volume ~2 x 1bps/lifetime of Universe

| Time interval | Data volume | Equivalent |
|---|---|---|
| 1 second | 100 MB | 1 linear metre of books |
| 1 minute | 6 GB | 1 Exabyte 8500 |
| 1 hour | 360 GB | 15000 trees-worth of paper |
| 1 day | 8.6 TB | US Library of Congress |
| 1 week | 60 TB | The NCAR MSS today |
| 1 month | 260 TB | The ECMWF MSS in 2002 |
| **1 year (100 days)** | **1 PB** | 3 years EOS data (2001) |
| 1 millenium (all) | 5 EB | All words ever spoken by humans |



# *The RD45 Project*

- Started in 1995 to address the issues of “object persistency” for the LHC experiments
- Strong focus on **standards**
  - Assumed that LHC would use C++
    - How does this compare with previous, Fortran-based solutions?
    - How can we adapt to changes, e.g. Java?
- Have identified a solution built on **ODBMS + MSS**
  - Project is still oriented towards the LHC
  - Solutions adopted by many other experiments
  - COMPASS (CERN): 400TB/year, 1999 on
  - BaBar (SLAC): 200TB/year, Phenix, STAR (BNL): 300TB/year each
- Currently setting up production services
  - production data ~100GB, test data ~1TB

# *Why Use Standards?*

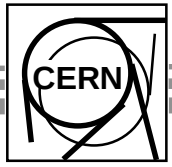
- Widely-used stds are an **essential** part of our strategy
- Cannot predict the changes between now and 2020
- Can be sure that major changes **will happen**
  - Stick with the rest of the pack
  - Do not use proprietary interfaces/solutions
- Even (especially) if you own the code, migration is a **slow, painful** process, typically measured in years

Which standards?

❶ C++, including STL-based collections

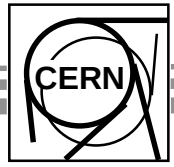
❷ Object Database Management Group (ODMG)

❸ IEEE MSS(?)



# *Persistent Object Manager: Choices*

- Strong feeling (before project started) that commercial solutions would not scale
- Investigated many potential solutions
  - language extensions (E, O++)
  - “light-weight” persistent object managers
  - OMG persistent object service
  - **full-blown Object Databases**
  - [ see Cattell’s book “Object Data Management” ]
- Strong preference for a **uniform** solution
  - or at least interface
- Candidates:
  - OMG POS, **ODMG**

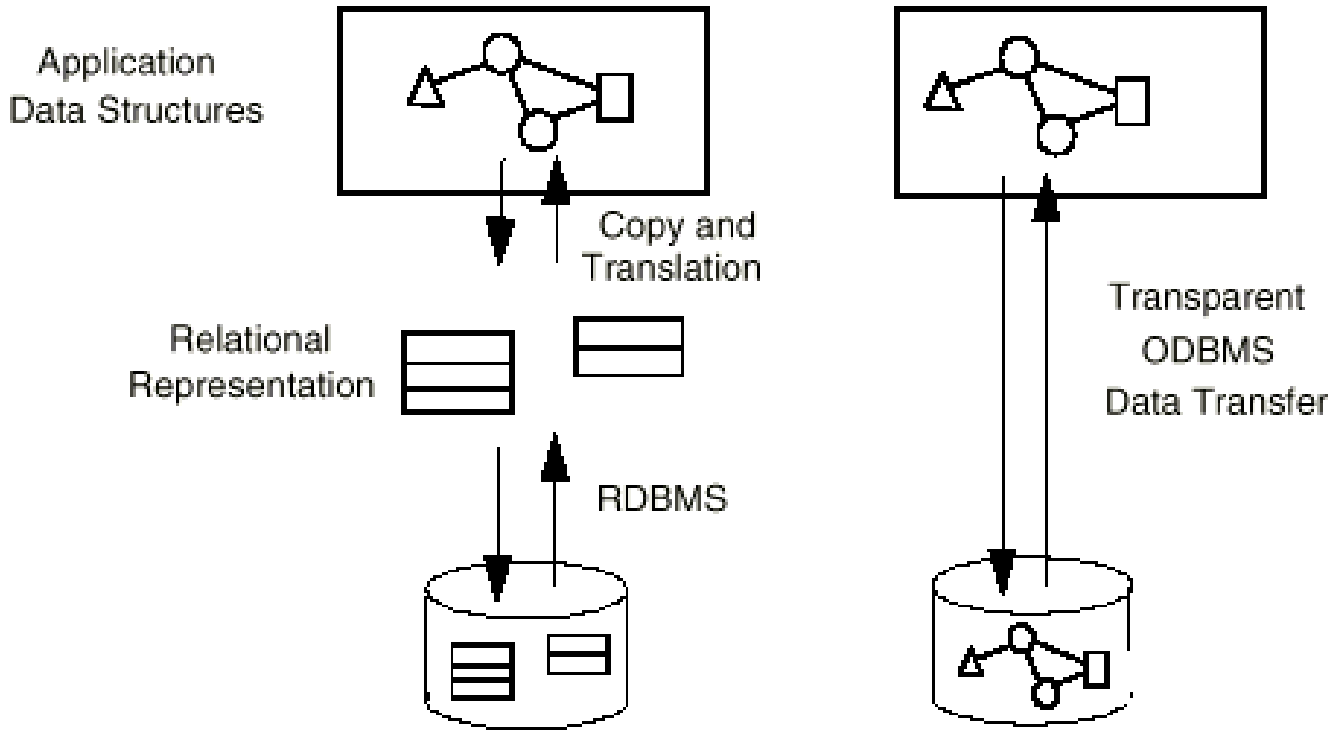


# *The ODMG*

The ***Object Database Management Group*** is

- a consortium of object-oriented database management system (ODBMS) vendors and interested parties working on standards to allow portability of customer software across ODBMS products
- ***"The programmer should perceive the binding as a single language for expressing both programming and database operations, not two languages with arbitrary boundaries between them."***

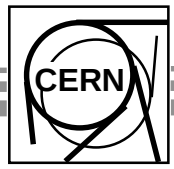
# *ODBMS vs RDBMS*



# ODMG vs Embedded SQL

```
   Init();        // initialise the db session
   startUpdate(); // start an update transaction
// create a new database (file)
   HepDatabaseRef  myDb = db("MyDatabase");
// create a new container in this database
   HepContainerRef cont =
     container("MyContainer");
   if (cont == 0 )
     fatal("could not find or create MyDatabase");
   for (short i=0; i<1000; i++)
      {
     // create a new event in my container
     HepRef(HepEvent) event = new(cont)
      HepEvent;
     if (event == 0)
       fatal("could not create a new event");
   }
```

```
EXEC SQL WHENEVER SQLWARNING GOTO
     1100
EXEC SQL LOCK TABLE FATMEN IN SHARE
     MODE
EXEC SQL SELECT * INTO :DBASE, :EXPER
     FROM FATMEN
   2       WHERE DATABASE =:DBASE AND
     EXPERIMENT =:EXPER
STMT1 = 'LOCK TABLE GNAMES_' //
     EXPER(1:LEXP) // ', FILES_' //
   1       EXPER(1:LEXP) // ', FXV_' //
     EXPER(1:LEXP) //
   2       ',VOLUMES_' // EXPER(1:LEXP) // ' IN
     EXCLUSIVE MODE'
CMD = 'STMT1'
EXEC SQL EXECUTE IMMEDIATE :STMT1
```



# *ODBMS Selection*

☹ There is no “best” ODBMS

- What are the requirements for your application?
- What are the priorities?
- What compromises can you accept?
- See Barry’s “Object Database Handbook”

Key requirements for CERN

- Scalability, scalability, scalability
- Performance
- Support for heterogeneous environments
  - languages, compilers, h/w etc.

☺ Some important simplifying factors...

# *Features of HEP Data*

☺ Data is essentially read-only

☺ Small number of concurrent users (1-100)

☺ Very low transaction rate

☺ Can accept data loss(!)

💣Data volumes and rates (up to 100PB, 1.5GB/s)

💣Fully distributed

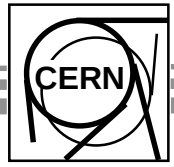
💣Project Lifetime (~25 years)

# *LHC Lifetime*

- Scheduled to enter operation in ~2005
- Will run until 2020+
- What will happen in the next 25 years?
- What has happened in the last 25 years?

At CERN:

- Introduction of computer terminals and phase-out of cards
- Installation of general-purpose networks
- Introduction of interactive services
- The death of the line-printer and “pyjama paper”
- Migration from mainframes to workstations to PCs
- Introduction of OO and move away from Fortran



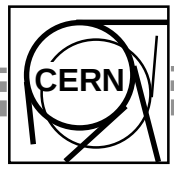
# *Can we use a commercial ODBMS?*

**Conventional Wisdom**

- Databases does not scale to PB range
- Besides, their too slow
- HEP requirements are just *different*
- We can't afford them anyway
- And even if we could, they won't be around in 2005

**RD45 results**

- ODBMS products with architectures that scale to PB range exist today
- (O)DBMS products sell on performance
- HEP requirements *are* different from the norm, but in some cases simpler! (WORM, low transaction rate etc.)
- We can't afford to build them
- industry standards are more likely to be around than HEP ones...

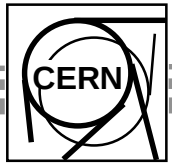


# *Building a multi-PB Database*

- Clearly cannot build a single DB of 100PB
- However, DBs of 1-10GB in production today
- Could we build a system out of multiple databases?
- What are the realistic limits on database size?
  - Filesystem limits (64-bit, no problem)
  - Disk/tape limits (50-100GB possible today)
  - **Transfer time: $10^2$ - $10^3$ seconds**
  - Effectively limits maximum filesize to around 10GB today

☹ Need 1M databases to reach 10PB

- but this is the same number of ***files*** that we are dealing with today...



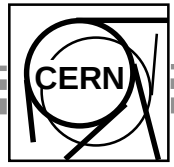
# *Architecture of Objectivity/DB*

- ~ODMG compliant; C++, Java, Smalltalk bindings
- Permits up to $2^{16}$ physical databases (i.e. files) in a single, logical database (federation)
  - tested up to 25GB/file (database)
  - tested up to limit of $2^{16}$ (small) databases
- Files may be distributed across database servers in LAN/WAN
- System/user data may be replicated
  - avoid single points of failure; bottlenecks

☺ Provides easily sufficient scalability to reach PB region

☹ Requires large (100GB) files for 100PB federations

☹ Cannot assume that all data will be on disk...



# *Multi-PB ODBMSs*

To extend the ODBMS to 100PB or more, require:

❶ An interface to a suitably scalable mass storage system

❷ Architectural changes permitting multi-PB federations without requiring very large files

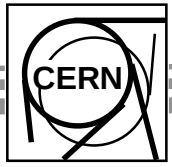
- required regardless of capabilities of MSS
- cannot assume that this will be installed on all systems, e.g. lap-tops

An interface has been designed and tested between our two product choices:

❶ Objectivity/DB (ODMG-compliant ODBMS)

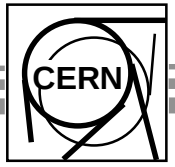
❷ HPSS

Interface developed by collaboration of Objectivity and SLAC



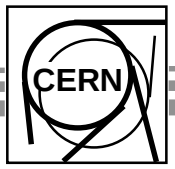
# *Architecture of Objectivity/DB*

- Federation: 1-$2^{16}$ databases
  - can be stored on different nodes in LAN/WAN
- Database: currently maps to a single file
  - [ containers, pages, slots, objects ]
- Each object has a unique 64bit Object Identifier
  - database, container, page, slot
  - database names stored in a catalogue
    - which can be replicated
  - user does not have to know ***where*** objects reside
- Objectivity client: object-aware
- Objectivity server: deals with files, pages etc.



# *Interfacing Objectivity/DB to HPSS*

- Objectivity/DB server uses a small number of I/O calls
  - *open(), close(), read(), write()*, etc.
- For each of these, an HPSS equivalent exists
  - *hpss_open(), hpss_close(), hpss_read(), hpss_write(),* etc.
- Essentially just “re-link” Objectivity/DB server against HPSS client API
- Such a “proof-of-concept” prototype was developed & announced at SC’97
- Currently being tested at Caltech, CERN, SLAC
- Expect a “product” by end-1998
  - numerous issues to be resolved, including performance questions related to use of “simple” HPSS API



ecycle Bin

hie's on NICE

hie's on AFS

nternet plorer 3

amie's files

Phone book

rtcut to 32.exe

Quick Search

Enter Keywords

**New!!**
**Order "A Technical Overview of Object Databases" video**
Click here to download a preview

# Objectivity ONLINE

- Corporate Information
- Products & Training
- Objectivity Events
- Solutions & Customers
- Other Resources

BUILT ON Objectivity

Objectivity, Inc. is the leader in enabling scalable object solutions, with over 140,000 Objectivity/DB server licenses deployed. Our customers develop software solutions for telecom, process control and new media applications.

## What's New !?

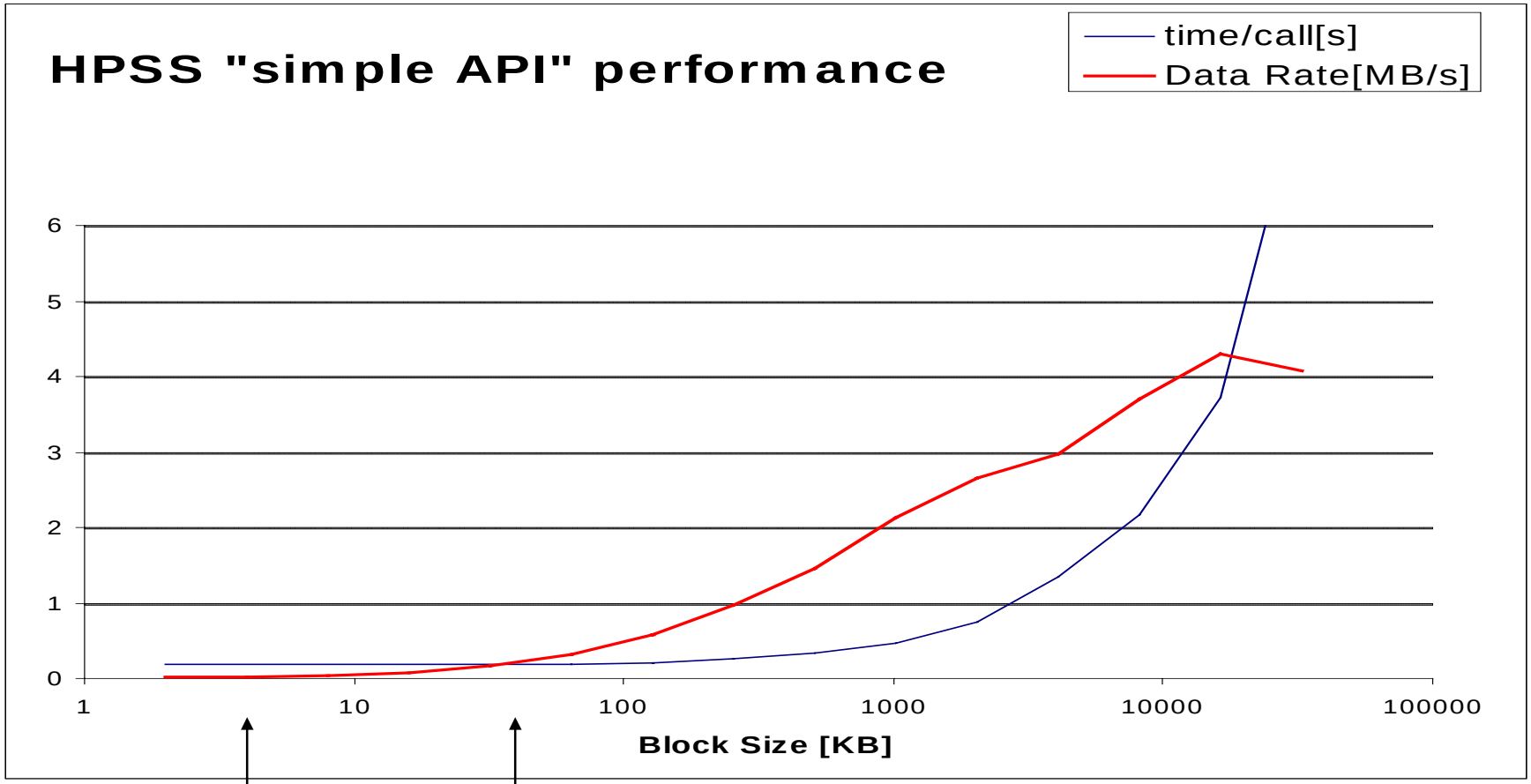
Objectivity support for HPSS will be showcased at the High Performance Networking and Computing show, SC97 in San Jose. Objectivity/DB's ability to support truly large databases complements the proposed HPSS solution to scalable storage.

SAN JOSE SC97 NOV. 15-21

Objectivity for Java delivers truly scalable solutions because the foundation is Objectivity's advanced distributed architecture which provides a single logical view of the federated database across multiple

Objectivity for Java

# HPSS Performance Measurements



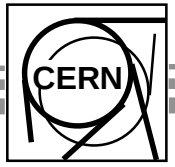

ODBMS page size

# *Objy-HPSS Interface*

Alternatives:

- Via the HPSS NFS interface,
- Using a future HPSS interface to DFS or DMIG,
- Using the "simple API" (the current interface),
- Using the "advanced API" (permits multiple blocks to be requested),
- Using a DB-faulting mechanism ("staging").

- For 98+, plan to use staging mechanism
  - DBs will be copied out of HPSS into Unix filesystems at open
- Can use existing hooks into Objectivity/DB to force stage of DB at open time
- Future plans include passing of hints from client to server, possibility of load-balancing across multiple servers etc.



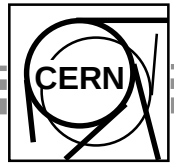
# *Accessing A Database*

- ***OO_FD_BOOT*** environment variable points to “boot-file”
- User “initialises” DB
- Can now access any object in any database in the federation
- Obviously, just iterating through the entire federation is not very convenient
- Can scan databases, containers, collections, look-up by name etc.

- Contains basic DB parameters
  - location of lock-server
  - journal-files
  - catalog & schema

```
int status = ooInit();
```

- DB host & filename determined from Object ID (OID)
- If file is not on disk, will be automatically restored by HPSS



# *Data Organisation*

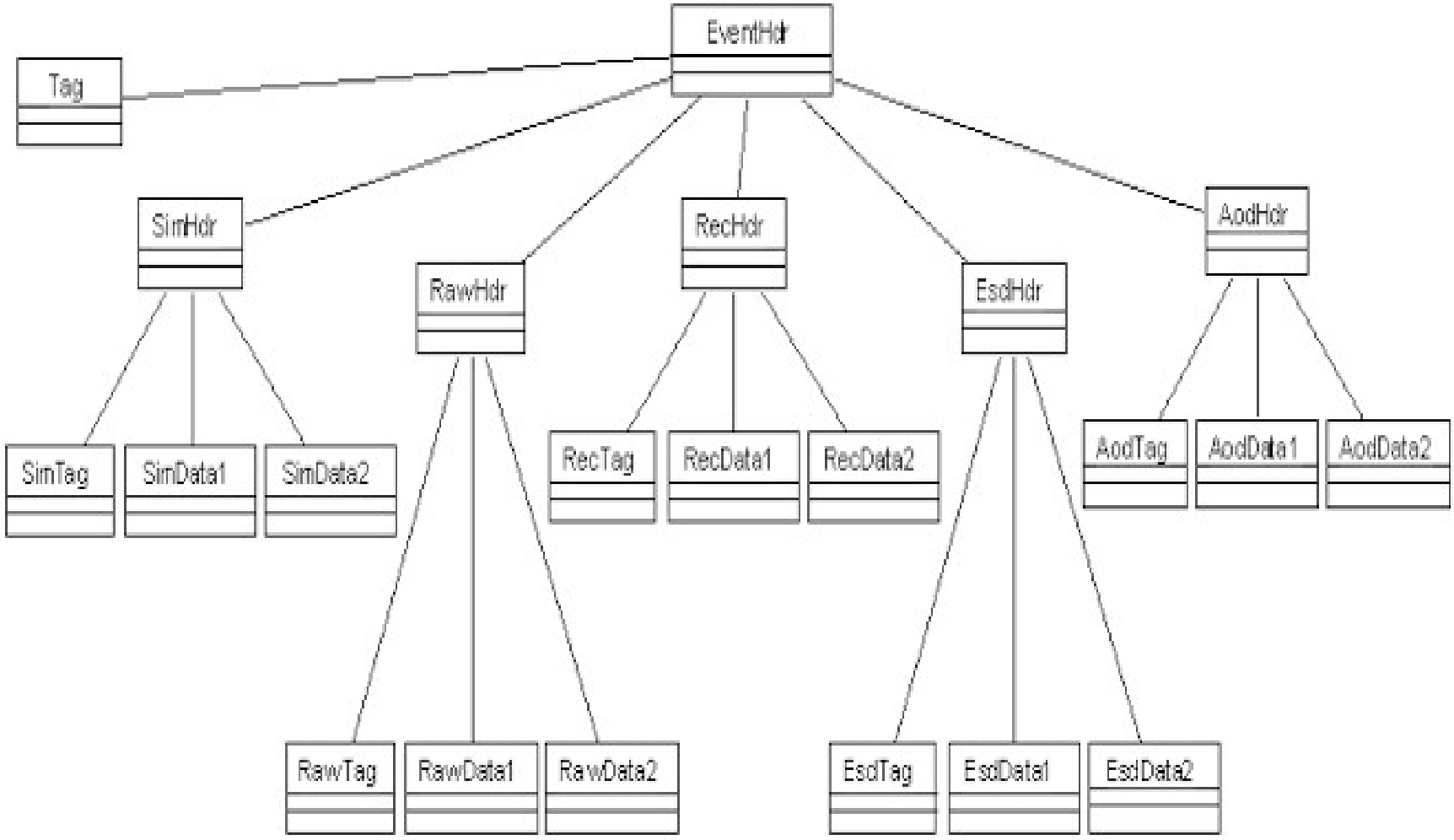
Physical

- Databases, Containers, Pages
- Convenient for data that will be accessed together
- Different “parts” of the same ***logical*** structure can have different physical clustering
  - e.g. infrequently accessed data is stored separately from “hot” data
  - clustering interesting ***parts*** of interesting events

Logical

- e.g. all events with certain characteristics
- may span multiple physical “containers”
- may be sparse
- may overlap / be interleaved with other logical collections
  - all events with 2 electron candidates
  - all events with 2 electrons & 2 μ

# *Example Event Structure (BaBar)*



# *Meta-Data*

Conventional definition:

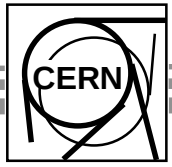
- Meta-data = “data about data”

Prefer:

- Meta-data = “data which makes other data usable”
- Meta-data for an LHC experiment ~TB
  - dominated by detector calibration information
- See significant benefits in using same system for both data & meta-data
  - can make logical associations between the two
  - can use different clustering strategies
    - meta-data permanently on disk

# *Short Term Plans*

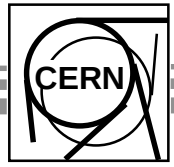
- Use of Objectivity/DB by existing experiments
  - Growing number of experiments at CERN and outside
  - Even adopted by Fortran-based experiments!
- LEP data archive
  - plan to archive some 30TB of LEP (1989-2000) data and keep for 20 years
  - understood that reprocessing will be a major effort
  - will “piggy-back” on LHC solution
    - Objectivity/DB, HPSS
- Plans for 1999+
  - COMPASS, NA45: ~400TB/year
  - similar data volumes at other labs (SLAC, BNL, …)



# *Comparisons with Existing Systems*

Significantly more powerful than existing HEP systems

- Consistent interface to *all* persistent data
- Data and meta-data handled by the same system
- Physical storage model de-coupled from logical model
- Applications much faster to build
  - e.g. calibration database: 1 student-year vs 10 expert-years
- Major performance and usability improvements seen
  - read only the needed objects
  - use meaningful predicates rather than semi-arbitrary bit-mask
- Much better integration with programming language
- Can handle >> data & provide enhanced functionality



# *Risk Factors*

- The survival of a given database vendor
- The survival of a given MSS system
- Neither can be guaranteed over a period of ~20 years!

- Production deployment in a fully distributed environment has yet to be demonstrated

- Best protection: use widely-used solutions for which there is a **demand**

# *Summary*

- We have demonstrated that a powerful, scalable data management solution can be built using “off-the-shelf”, standards-influenced components
- We believe that commercial Object Database Management Systems are a viable solution for bulk data storage
- A combination of an ODBMS + MSS can handle data volumes well into the PB region
- Parts of the system have been demonstrated in production
- Full production is expected in 1999
- By 2000, expect > 1PB in Objectivity/DB + HPSS