# Data Persistency Solution for LHCb

- Motivation
- Data access
- Generic model
- Experience & Conclusions

M.Frank LHCb/CERN - In behalf of the LHCb GAUDI team

# Motivation

- **Physics software should be independent of the underlying data storage technology**
- **Data of different nature has to be accessed**
  - Event data, detector data, statistical data, …
  - The access patterns differ
  - The data set size varies from several Mbytes to 1 Pbyte
  - Legacy data was written in ZEBRA format
- **It is unclear how these data will be stored**
  - Locking into one technology may be a disadvantage

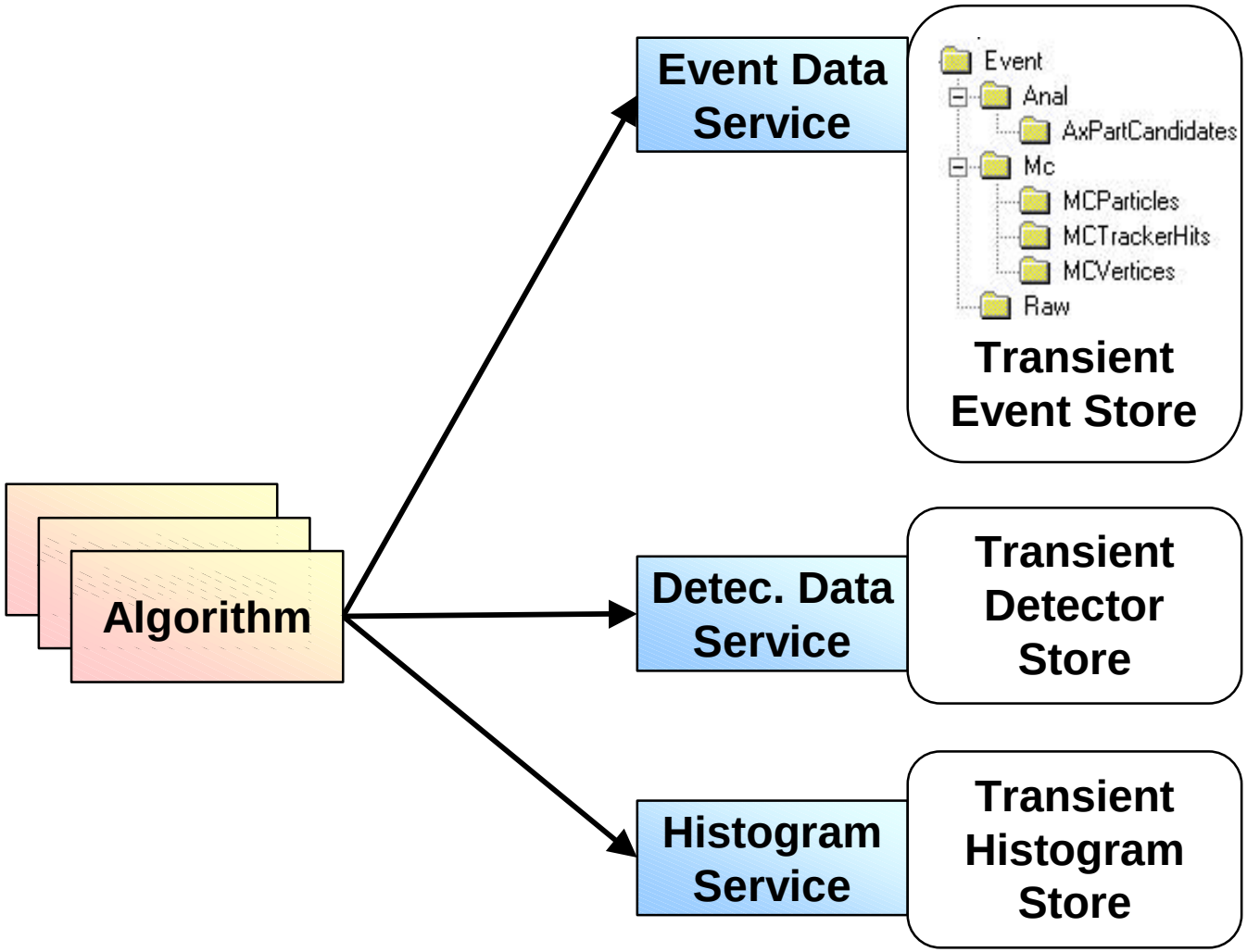


# Strategy

- **Transient data representation is separated from the persistent data representation**
  - Each representation can be optimized separately
  - Transient representation can be used to convert to any other representation
- **Minimize coupling between algorithms and the transient data**
  - Algorithms see only transient data
  - Transient data items are not intelligent
  - Algorithms post and retrieve transient data from a "black-board", the data store

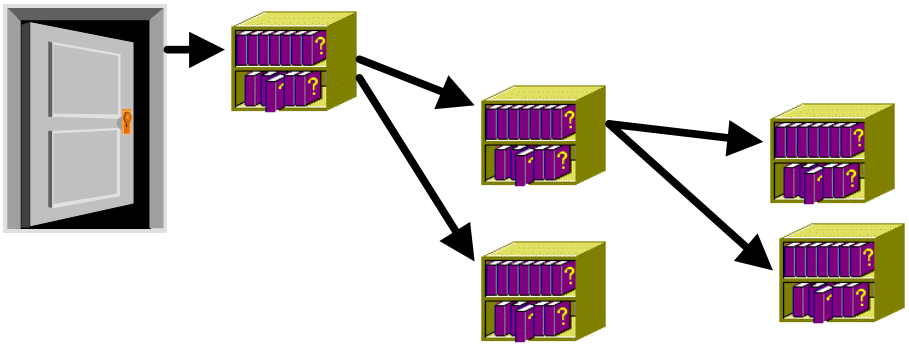


# How are Data Accessed?



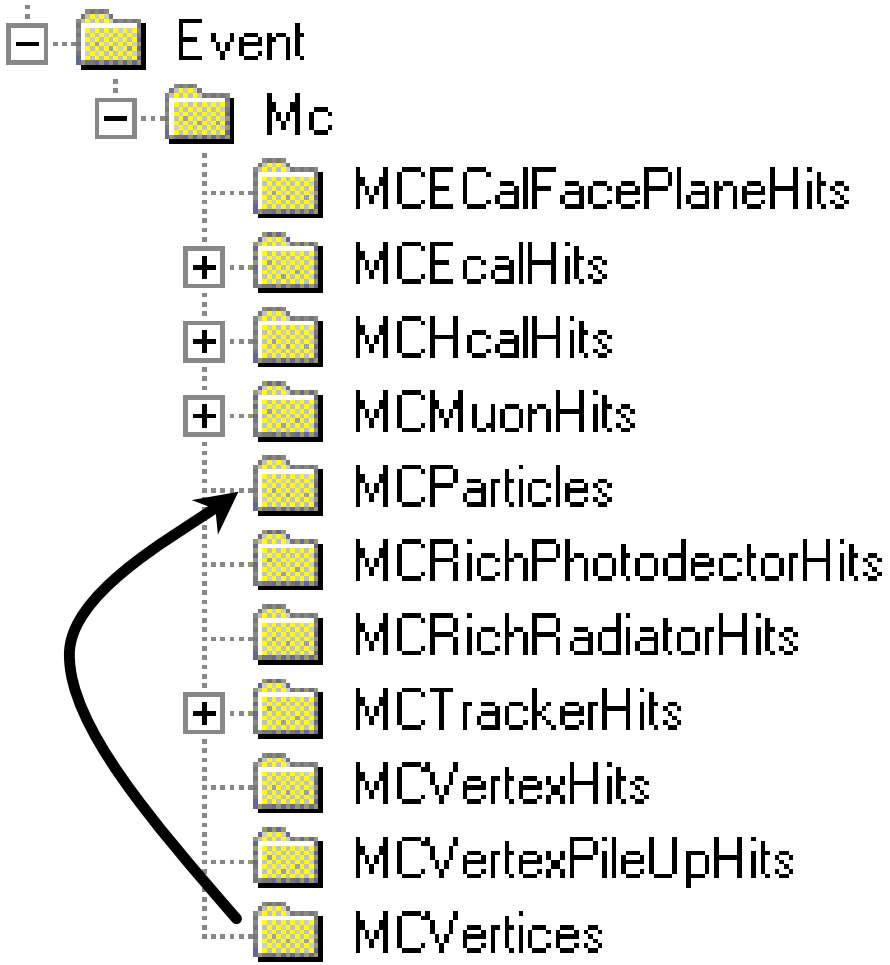
# Functionality of Data Stores

- Manage objects of **similar lifetime**
- Manage data objects like a librarian
  - Clients store objects
  - Other clients pick up objects when needed
  - Retrieve object collections



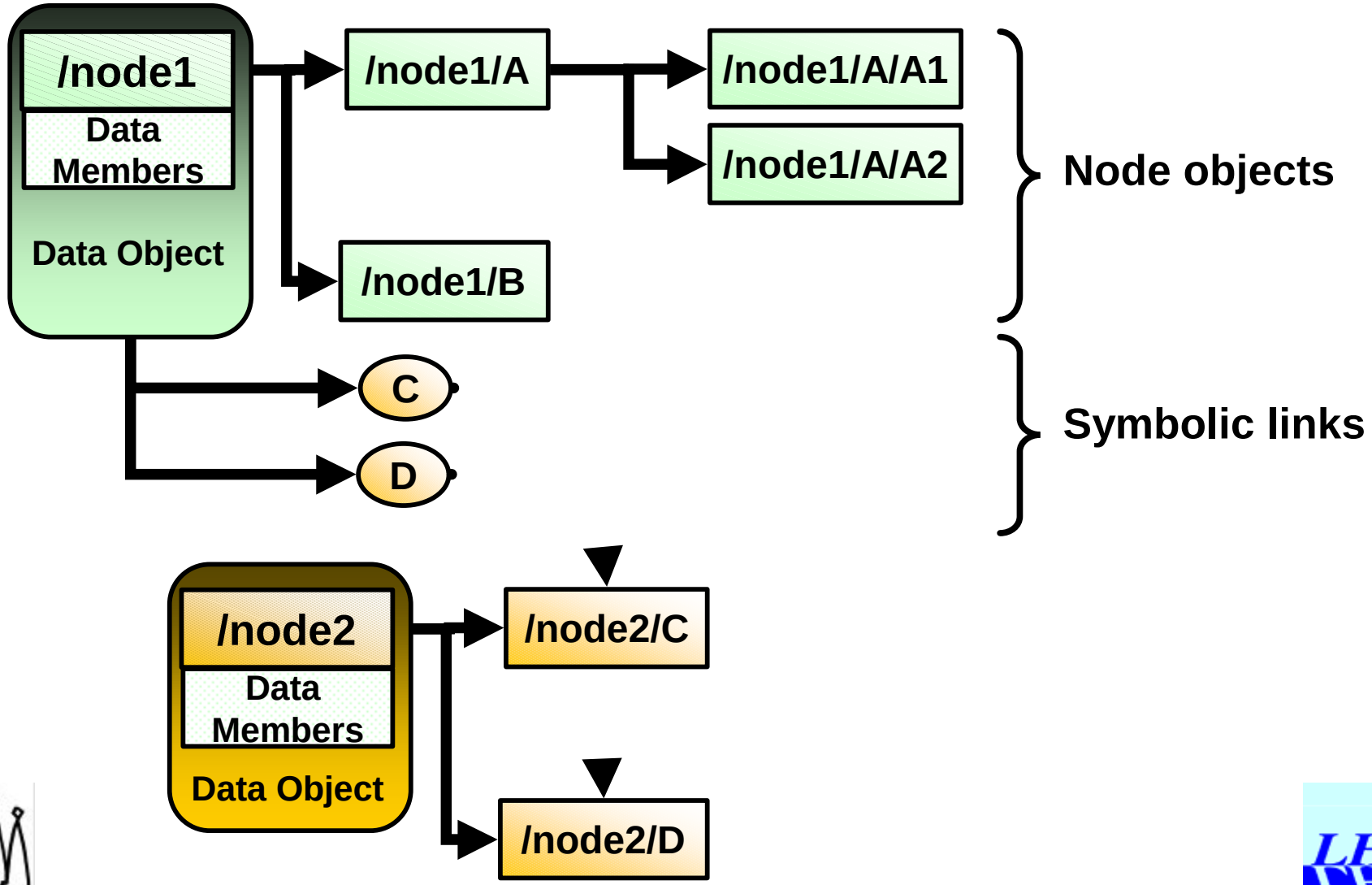
- Manage **ownership**
  - Does cleanup

# Structure of the Data Store

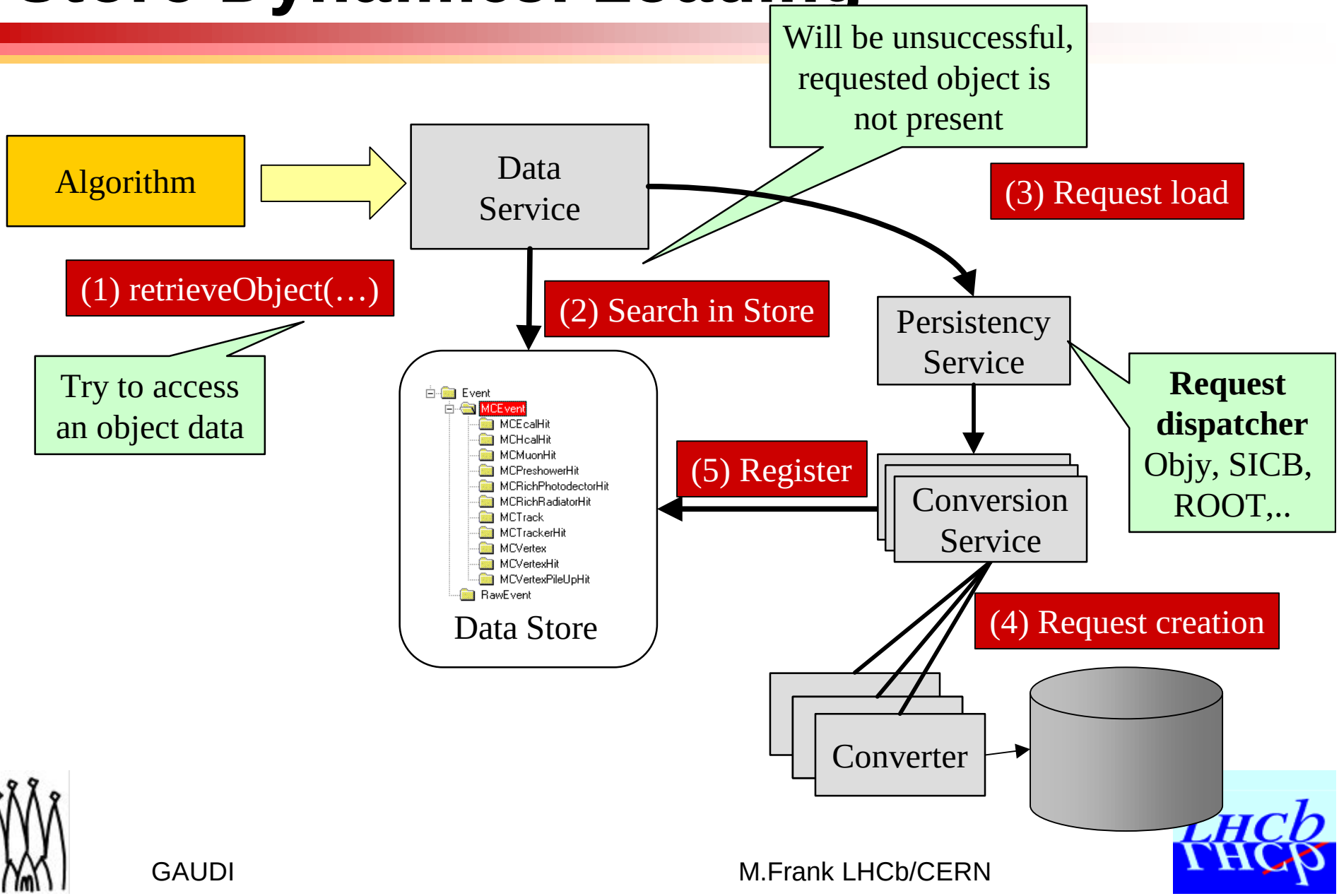



- Tree - similar to file system
- Identification by logical addresses:
  "/Event/Mc/MCParticles"
- Tree node
  - has data members (payload)
  - contains other node objects (directory structure)
- Browse capability

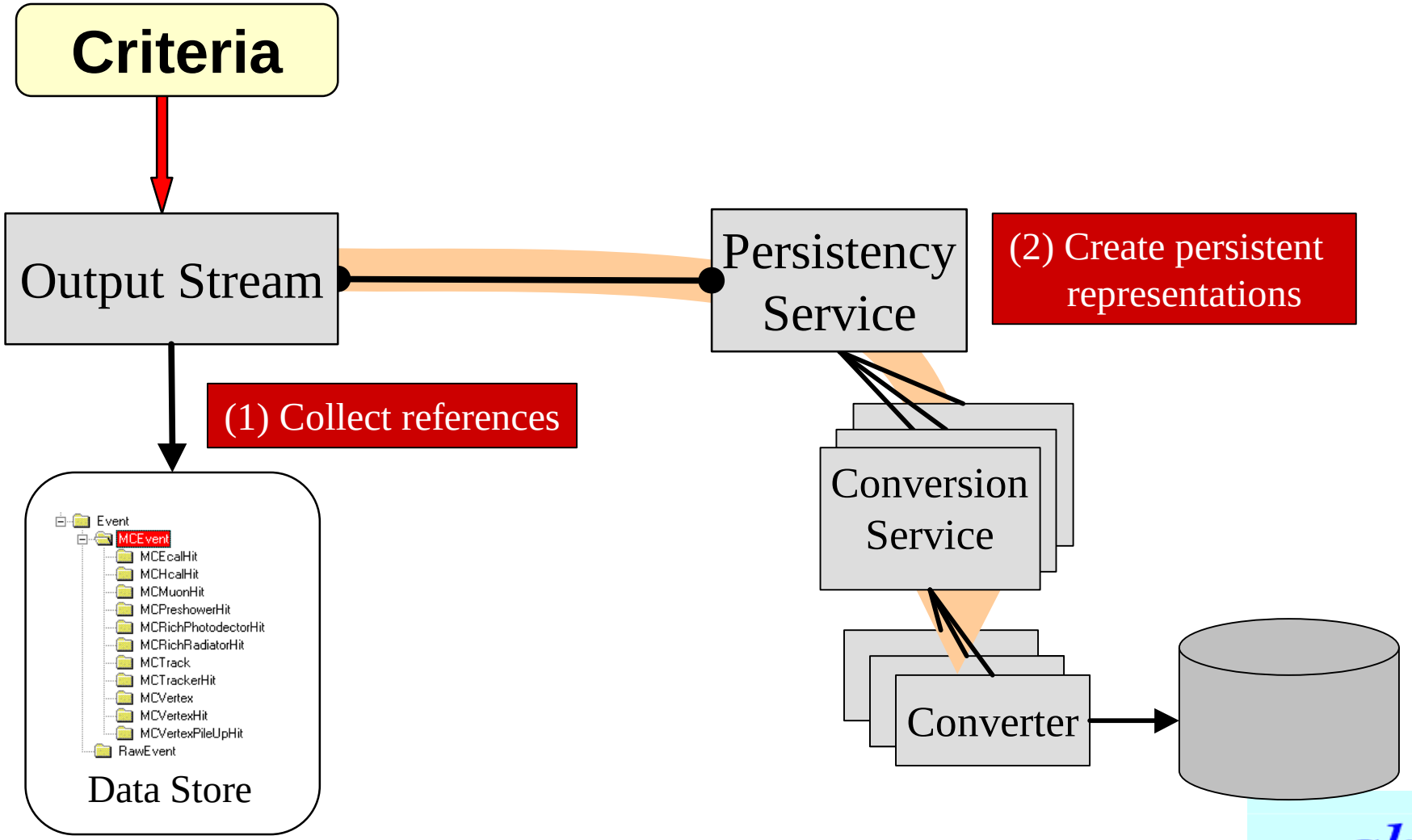
# Layout of the Data Object



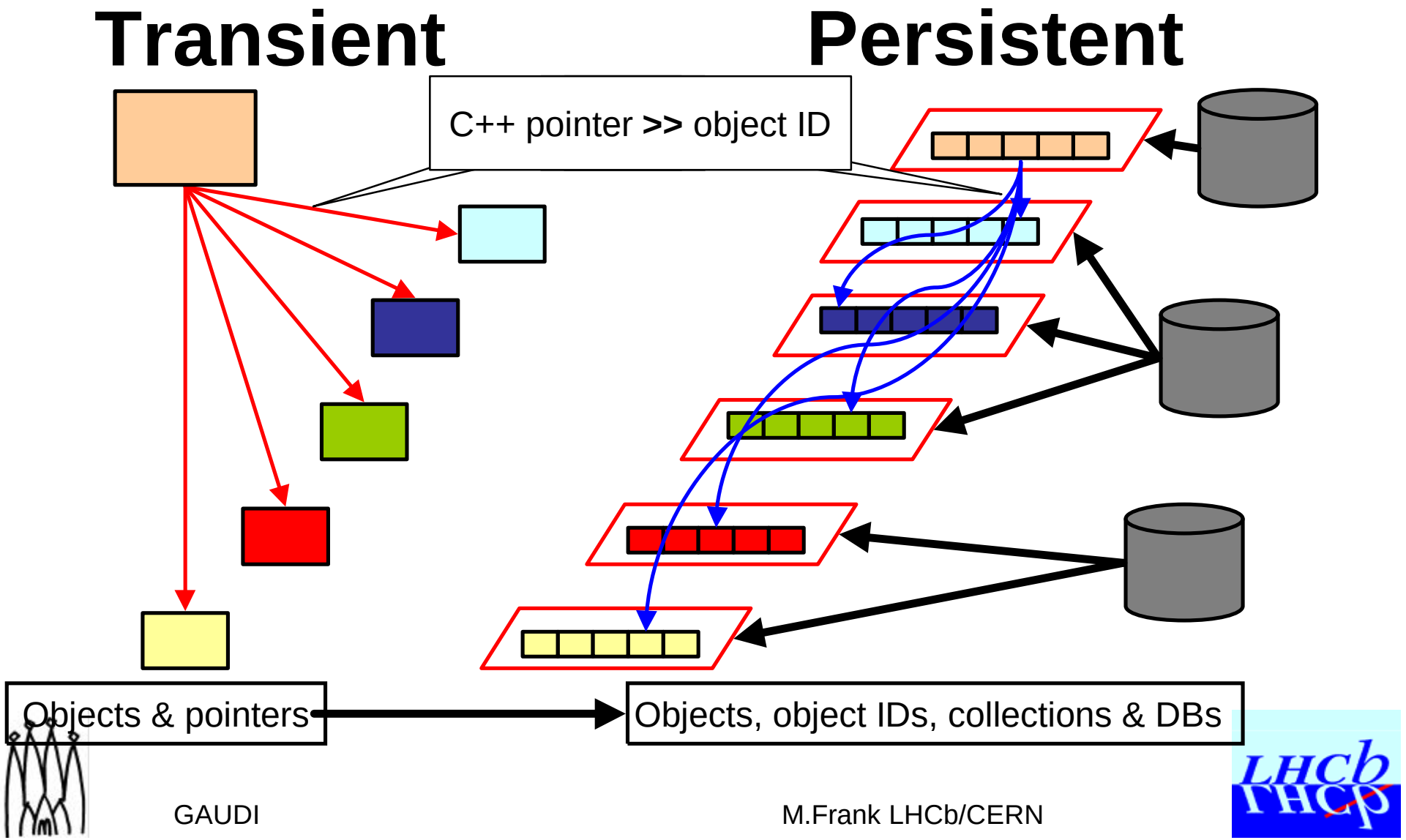
# Store Dynamics: Loading



# Store Dynamics: Storing



# Generic Persistent Model

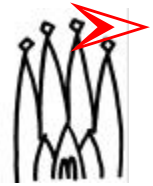


# Database Technologies

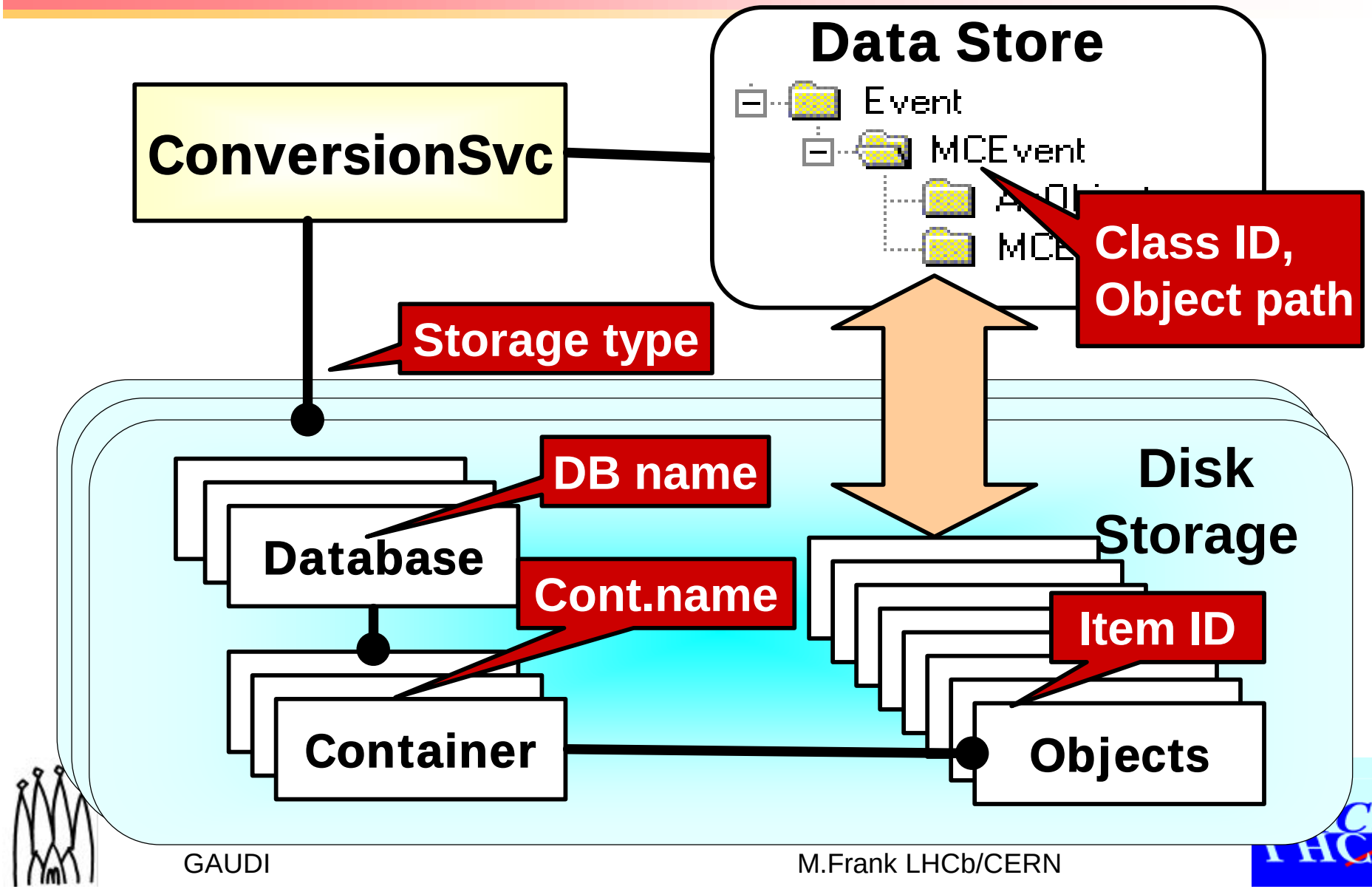
- Identify commonalties and differences
  Necessary knowledge when reading/writing

| | Generic | ZEBRA | ROOT | RDBMS | Objy |
|---|---|---|---|---|---|
| Write | Database | File | File | Database | Database |
| | Collection | Bank | Tree/Branch | Table | Container |
| | Item ID | Record # | Event # | Prim.Key | |
| Read | Database | As for writing | | | OID |
| | Collection | | | | |
| | Item ID | | | | |

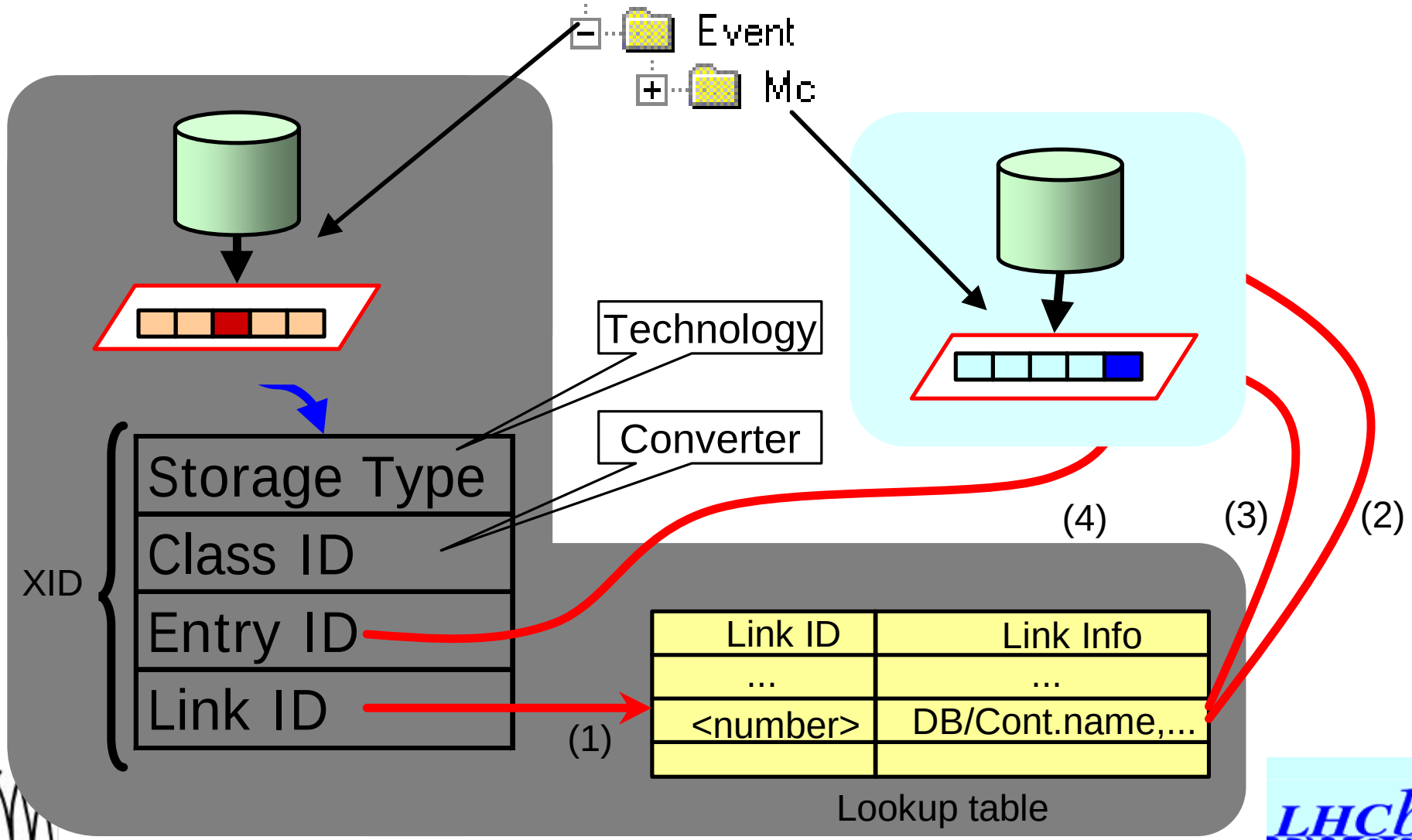
- RDBMS: More or less traditional
- Objy is different: How to match ?



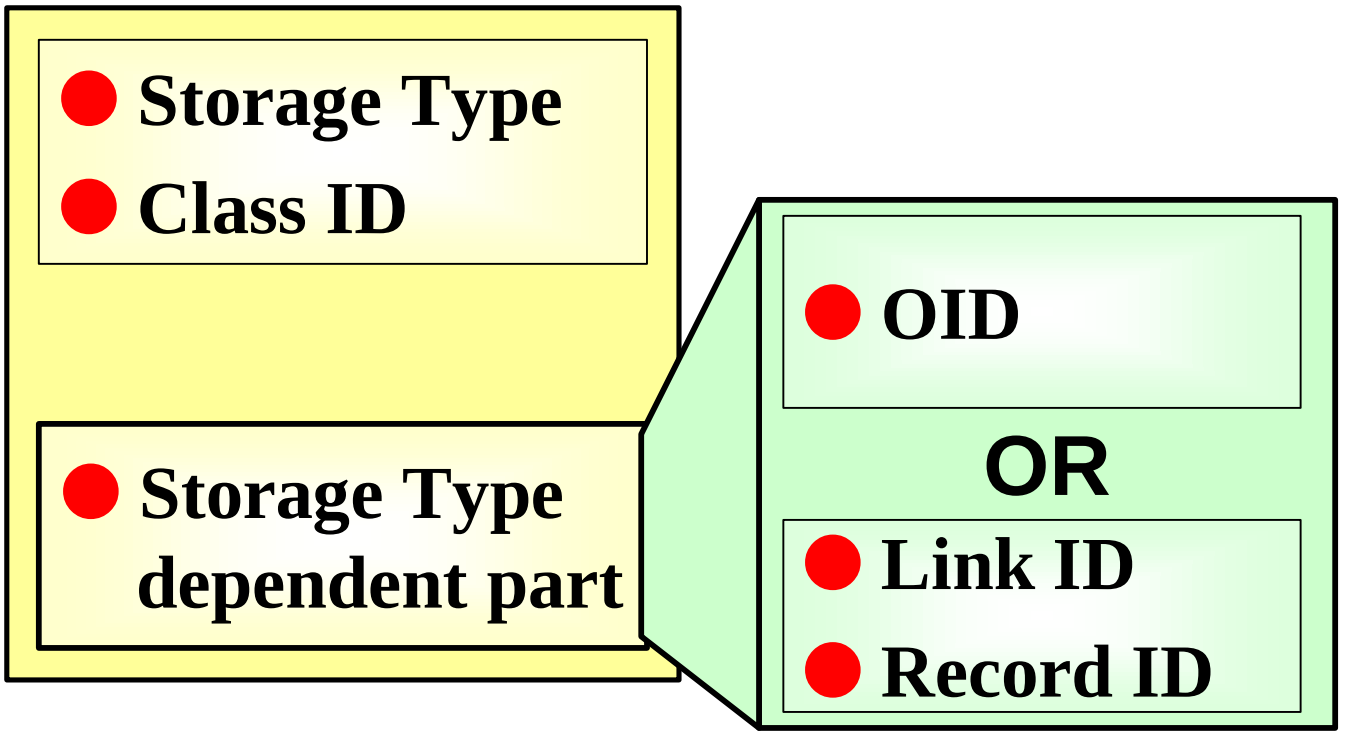
# Generic Model: Assumptions



# Generic Model: References



# Generic Model: Extended Object ID




Objectivity

ZEBRA
ROOT
RDBMS

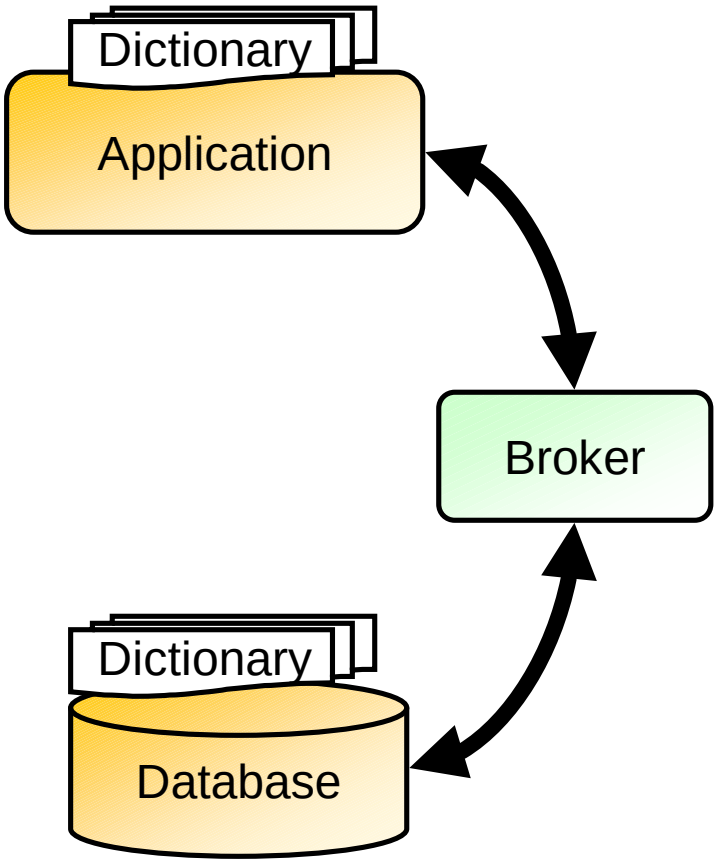
# Experience & Conclusions

- It is possible to write physics data without knowledge of the underlying store technology
- Our approach can adopt any technology based on database files, collections and objects within collections
  - ZEBRA, ROOT, Objy and RDBMS
  - We are able to choose technologies according to needs
- Overhead of transient-persistent separation looks manageable

**http://lhcb.cern.ch/computing/Components/html/GaudiMain.html**



# Object Evolution




- Handling of dictionary discrepancies between the application and the database
- "Generic" handling?
  - default values ?
- Architectural problem
  - Handled inside "Converters"
  - Better chance to supply reasonable values