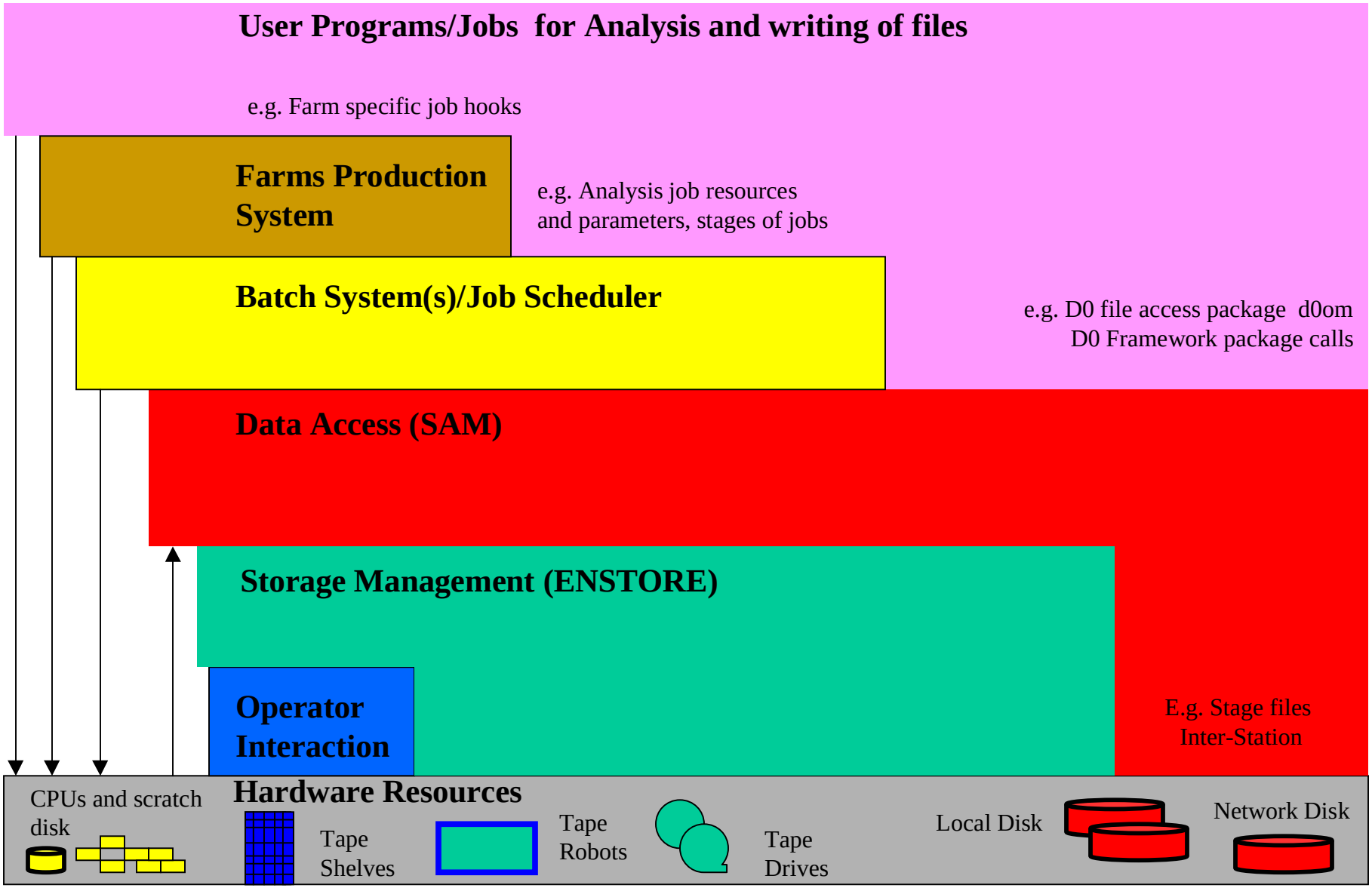
COMPONENTS OF SAM - Outline

- Interfaces with other layers
- Components and Software Architecture
- Implementation choices
- Database, Database Server
- Logger, Information Server
- Station, Project Master, Stager
- Optimizer and Resource Management

Components of SAM

- Layers
  - Interfaces between SAM and other layers
- Functional Components
  - the functional components of the SAM data access layer
- SAM Software Architecture and interfaces
  - an architecture for implementing software components which provide the functionality
- Implementation choices for the Prototype
  - more layers - this time implementation layers isolating technology such as particular database used, particular type of stager used, particular batch system to be used

Software Layers for Data Management and Analysis



What SAM is expecting of the Storage Management Layer

Cataloging & Database fns.

- primary file/ volume map
-access to file namespace and volume info
-status and history info

Specification and Control of Tape Storage Locations

Access and Control of Parameters which govern the functional behavior of the system (e.g. width of tape writes, file families, etc.)

Management of the Robot resources, including error recovery and tracking

Movement of files between users machine/local disk and tape in robot

Operational Procedures to Run and Manage the Robot, tape drives and "Databases"

What SAM is expecting of the Storage Management Layer

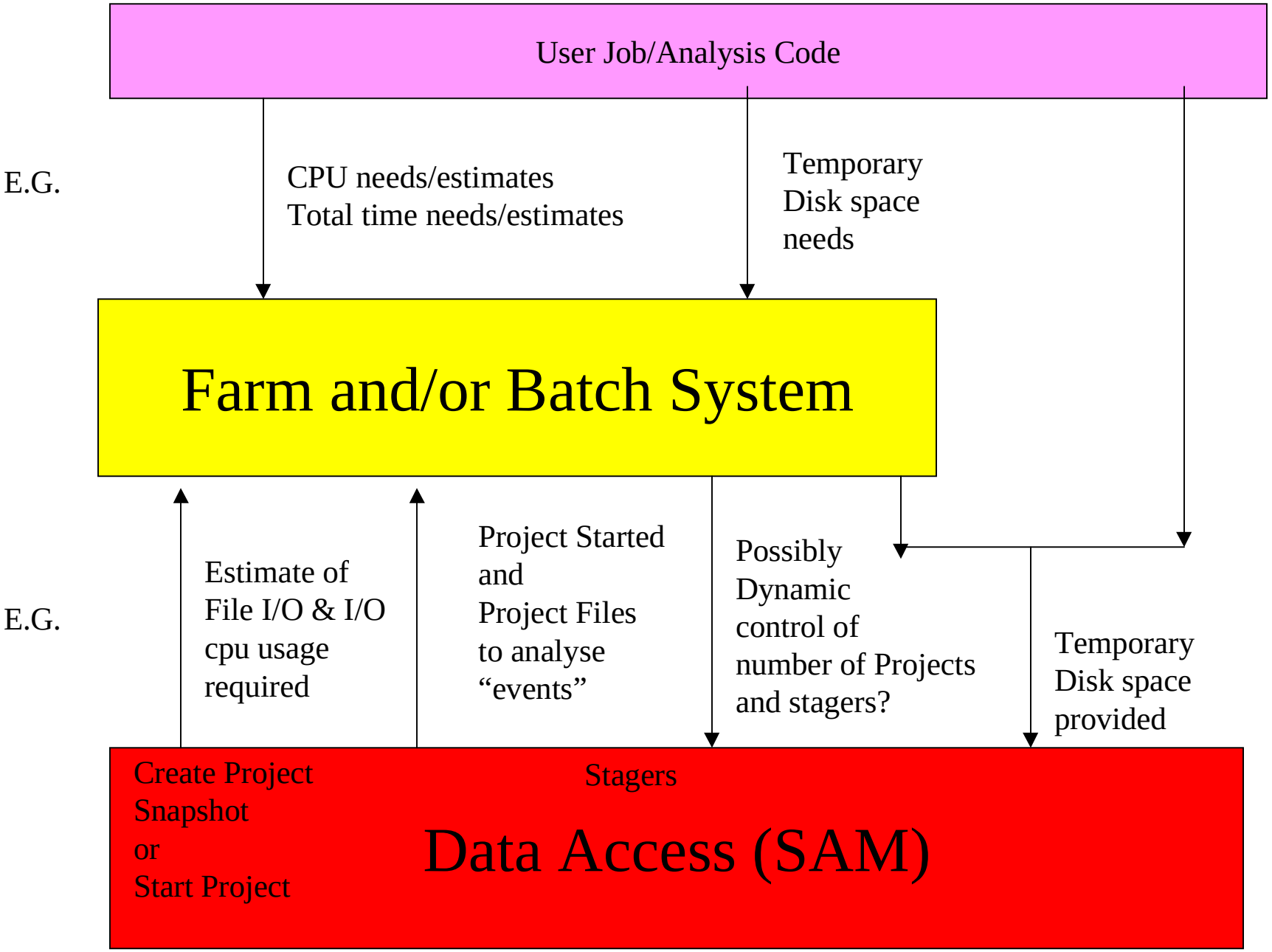
See detailed list of functional requirements and interface specification off
of SAM home page
http://runiicomputing.fnal.gov/sam
(or http://www-d0.fnal.gov/~white/sam
or http://www-d0.fnal.gov/~lueking/sam

What a Batch System and/or Farm Production Layer might expect of SAM

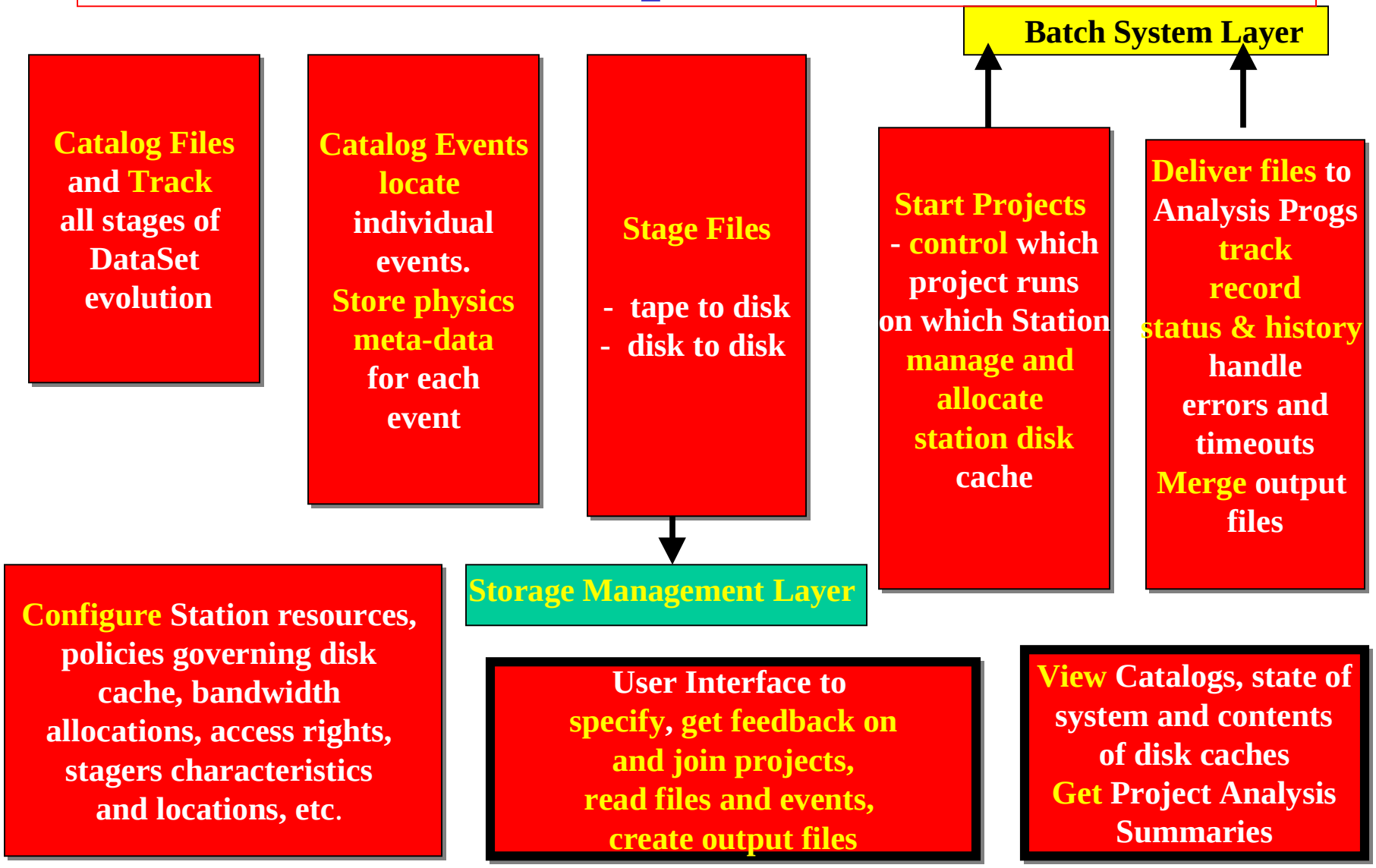
- SAM relies on a set of distributed server processes to do its work.
- These expect to allow/disallow Projects (to deliver data to one or more analysis programs) to run on a station according to policies, semi-static disk allocations and semi-static configurations held within SAM (in the config database).
- A Batch system, such as LSF (or its Job Scheduler) component could use “events” from the SAM layer as one of its parameters for spreading the load of jobs across systems and controlling the number of and priority of analysis jobs

===>> more later on Resource Management

Resource Management Interfaces (TBD)



Data Access Layer - SAM Functional components

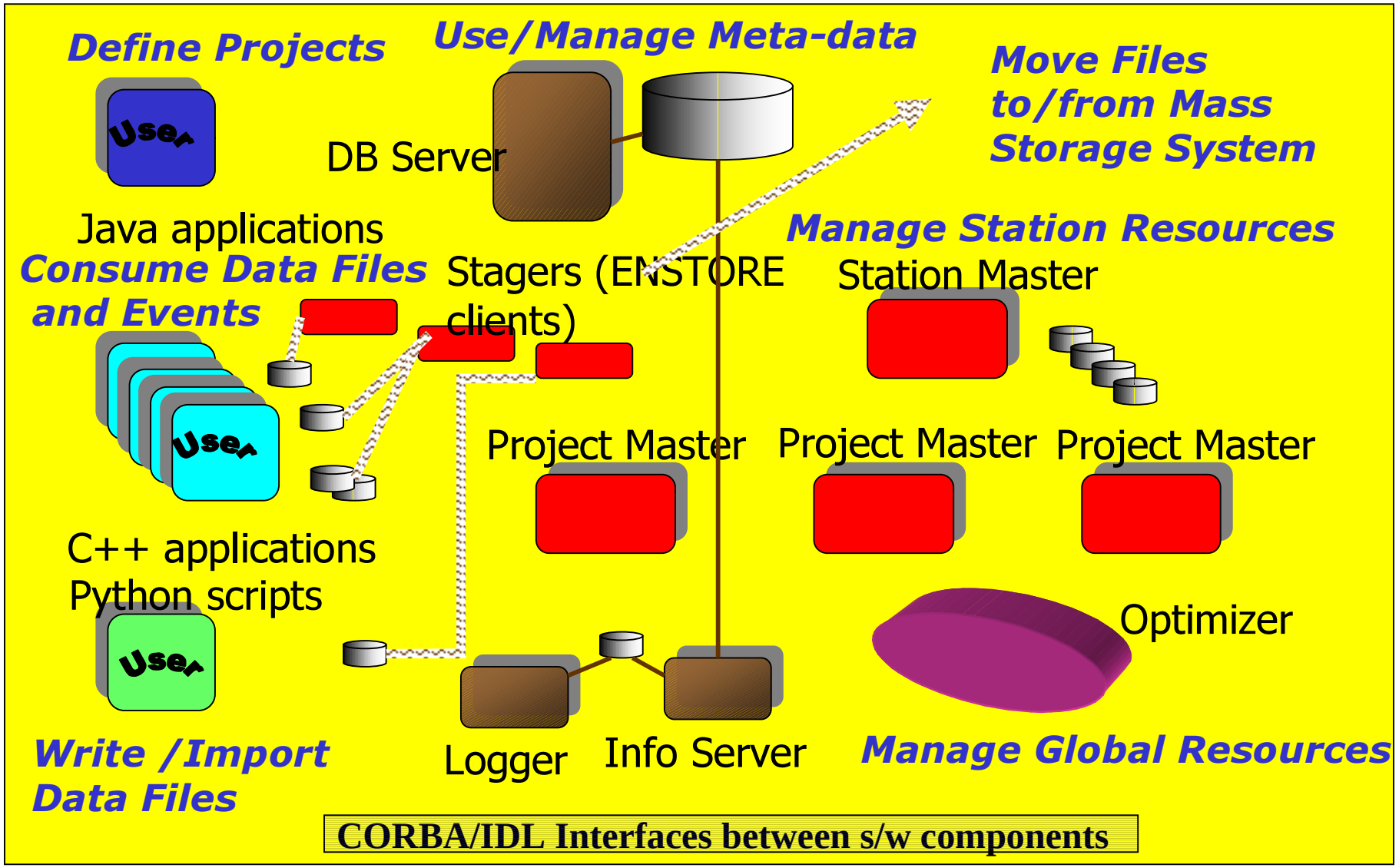


Functional Components <--> Design of a System

- Catalog, Track, Keep History, Record status, Configure, Record, Provide summary --> Database Server (permanent info) [1 or more], Logger (transient info) [1]
- Start Projects, Configure, Check access rights, Maintain Station disk cache inventory --> Station Master [1 per Station]
- Register Consumers, deliver file names, handle errors, allow users to join/rejoin projects, handle output files, manage assigned disk and staging of files --> Project Master [N per Station]
- Stage Files --> Stager or encp client [>=1 per physical disk cache] (interface to Enstore)
- Optimize and Prioritize staging work for Enstore --> Optimizer [1]
- User interfaces to specify projects, read/write files, configure system, view system --> interfaces to other components providing these services and InfoServer [1] to pre-digest information

The verbs of the functional components map to methods (Interfaces) of S/W components

Software Components of SAM



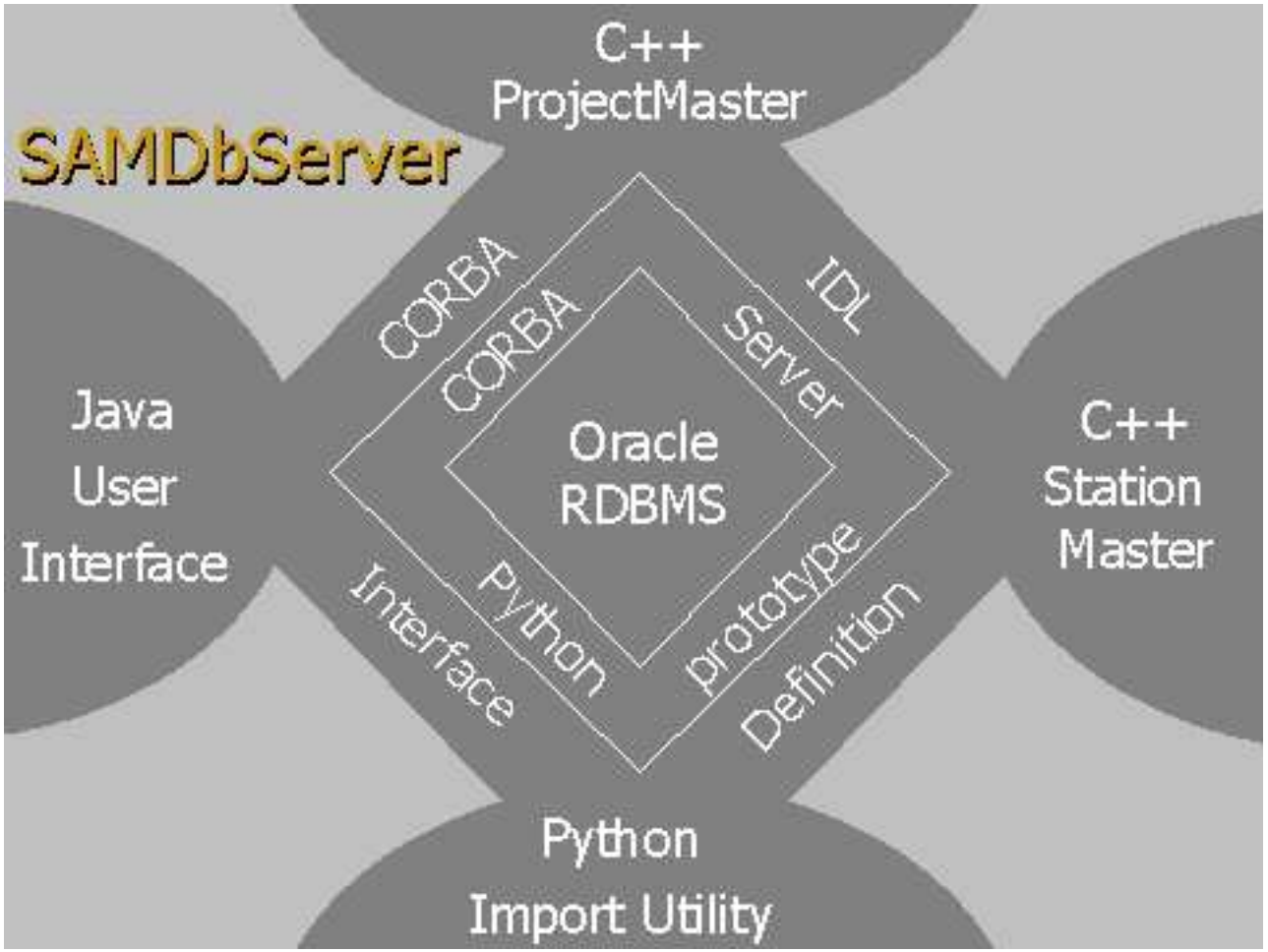
Component Framework

- Client/Server - one component (acting as Client) asks another component (acting as Server) to perform some task. The status and results are returned to the Client component.
  - One Server may service many clients simulataneously and maintain state (multiple threads, queues) if in turn that Server has to ask another Server for services
- Multiple Analysis programs asking for Files and Data proceeds asynchronously to getting files staged to disk
- High Level protocol between components to arrange for Enstore file staging jobs to be 'batched' up and to be presented in optimized order for a tape volume.

CORBA to implement client/server framework

- Define the Interfaces between components --- what does one component want from the other. IDL Interface Definition Language
- Define the data structures needed for one component to talk to another - part of IDL
  - take a look at some of the IDL in the code tree off of SAM web page
- Use an IDL compiler to generate a huge amount of the Server and Client code in any one of 3 languages - Python, C++ or Java
- Implement the most complex Server (Project Master) (which must handle multiple clients, maintain state, handle timeouts,etc) using multiple threads of code

Software Architecture - example of one Server and its interfaces



Other Technology used in Prototype

- Oracle Designer tools - to design schema of database and generate sql for database
- Java development environments and Java JCLASS library for graphs
- 3 different free-ware CORBA products (ORBacus, ACE-TAO, Fnorb)
- Oracle Relational database

SAM Event, File, and Processing Database

- Overall goals and strategy
- Database server
- Database Schema
- Import and export

Goals and Strategy

Goals and Strategy

- **Provide a schema which satisfies all of the processing, re-processing, and bookkeeping problems encountered in Run I.**
- **Provide information at the file and event levels.**
- **Build the database in a solid, commercial, relational database to allow us to use the best tools and features, available, and to achieve maximum reliability. Oracle 8 is being used for on-site.**
- **Build in flexibility to employ non-commercial database products if required for off (fnal)-site compliance.**

Goals and Strategy

- **Provide convenient import and export mechanisms.**
- **Coordinate the information needed to perform reproducible physics analysis and easy access to trigger, luminosity and related information.**
- **Provide integrated access to the on-line database, run conditions, calibration, alignment, trigger configuration, et cetera.**

The Database Server

Database Server

- **Represents a “firewall” between the database and the many clients.**
- **Allows serving many clients with only one database login. Makes transactions quicker.**
- **CORBA enables us to employ the same interfaces for clients written in many different languages:**
  - User scripts in Python, Perl, Java…
  - User project front-end in Java.
  - Import scripts in Python.
  - Station and project masters in C++.

Database Server

- **Changes in the db schema can be put into the server without affecting the many client codes.**
- **The Oracle db can be replaced by mSQL, mySQL, ascii file, et cetera by changing the db server. No modifications to the clients are required.**
- **Currently the db server is implemented in Python for rapid development features. May change in the future.**

Database Server - CORBA

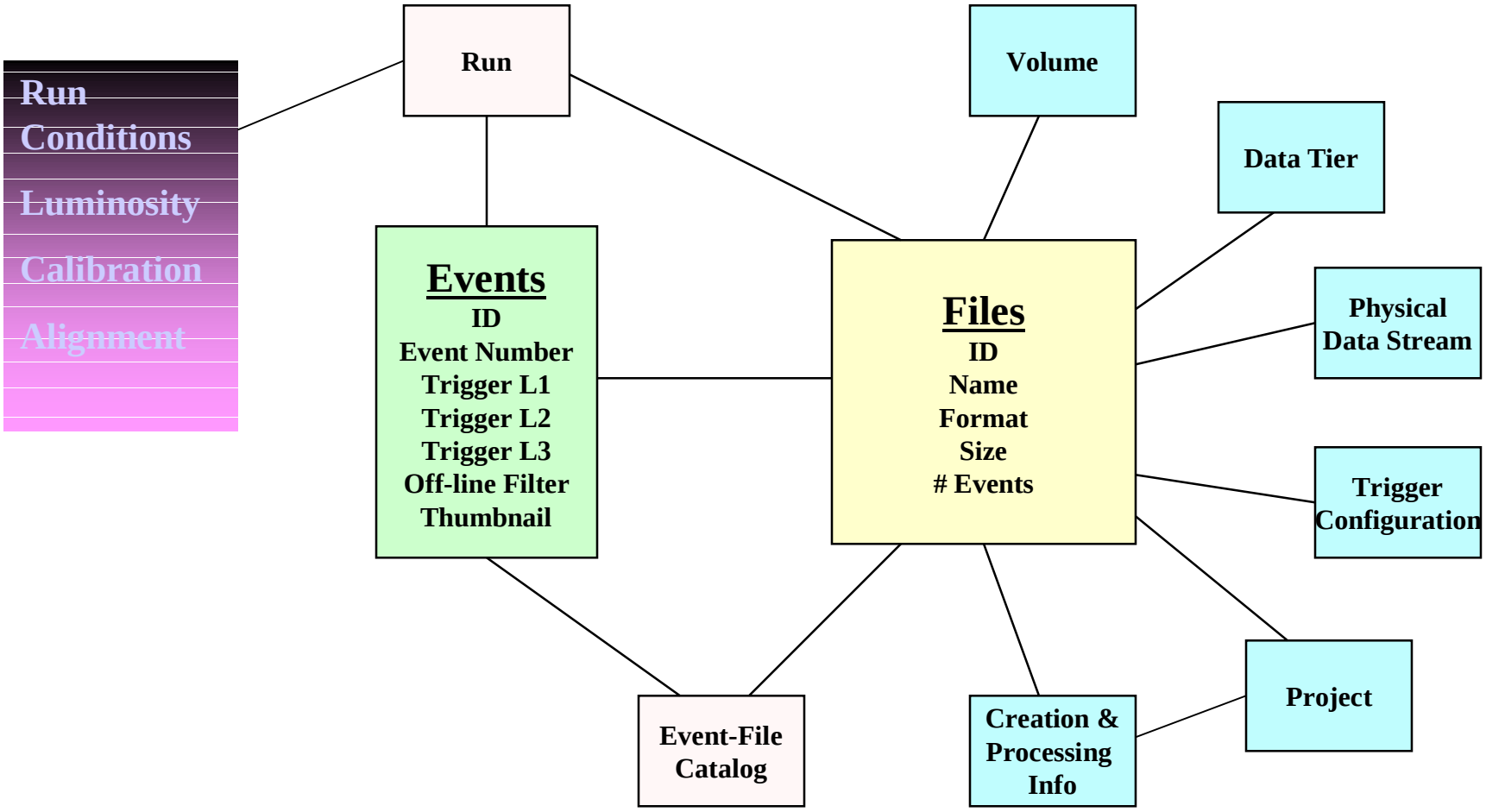
- All interfaces to the db server are written in IDL (Interface Definition Language).
- Examples…
  - http://runiicomputing.fnal.gov/sam

The Database Schema and Functionality

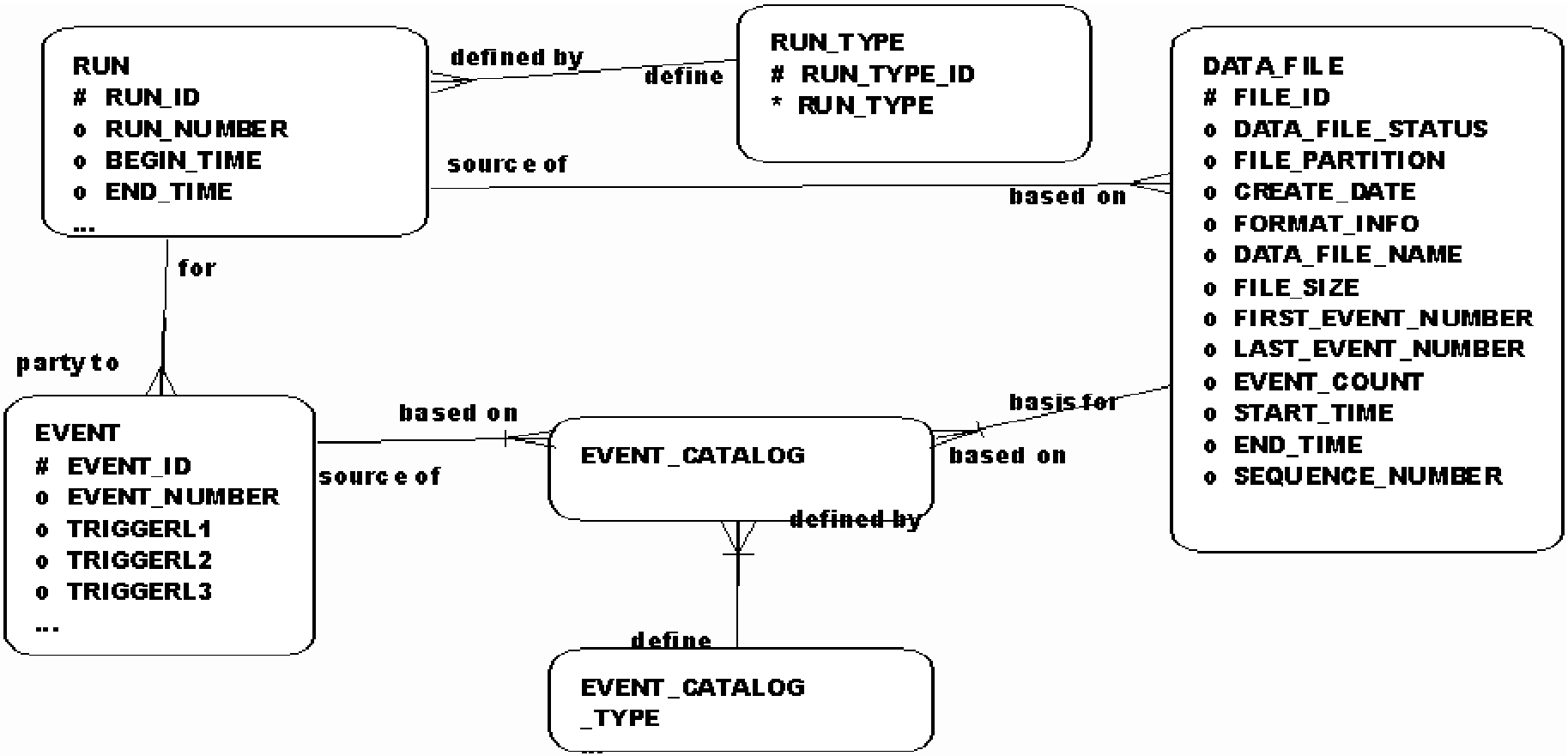
Database Schema

- **Database Schema**
  - On-line
  - Event Catalog
  - File Catalog
  - Physical volume, etc.
  - streams, triggers and luminosity
  - Processing
  - consumers
  - projects
  - processing chains

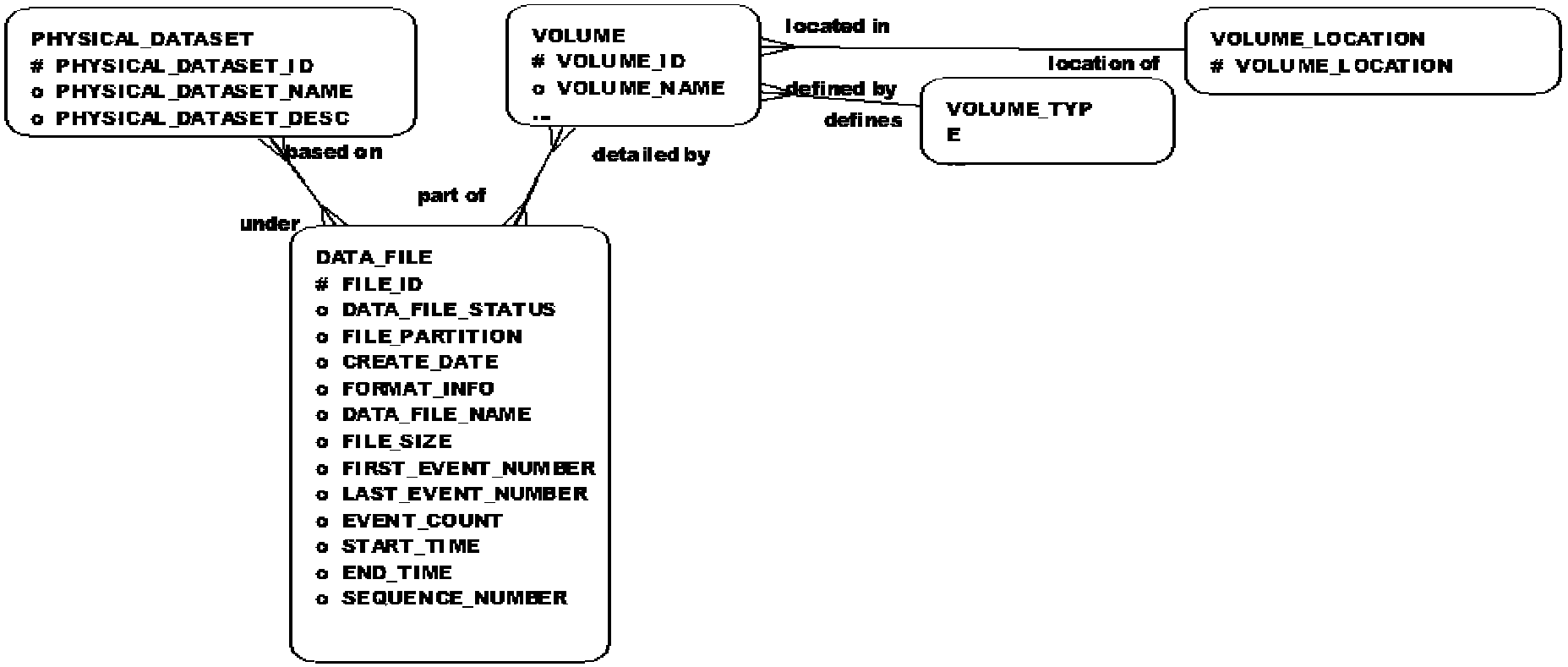
Simplified Schema



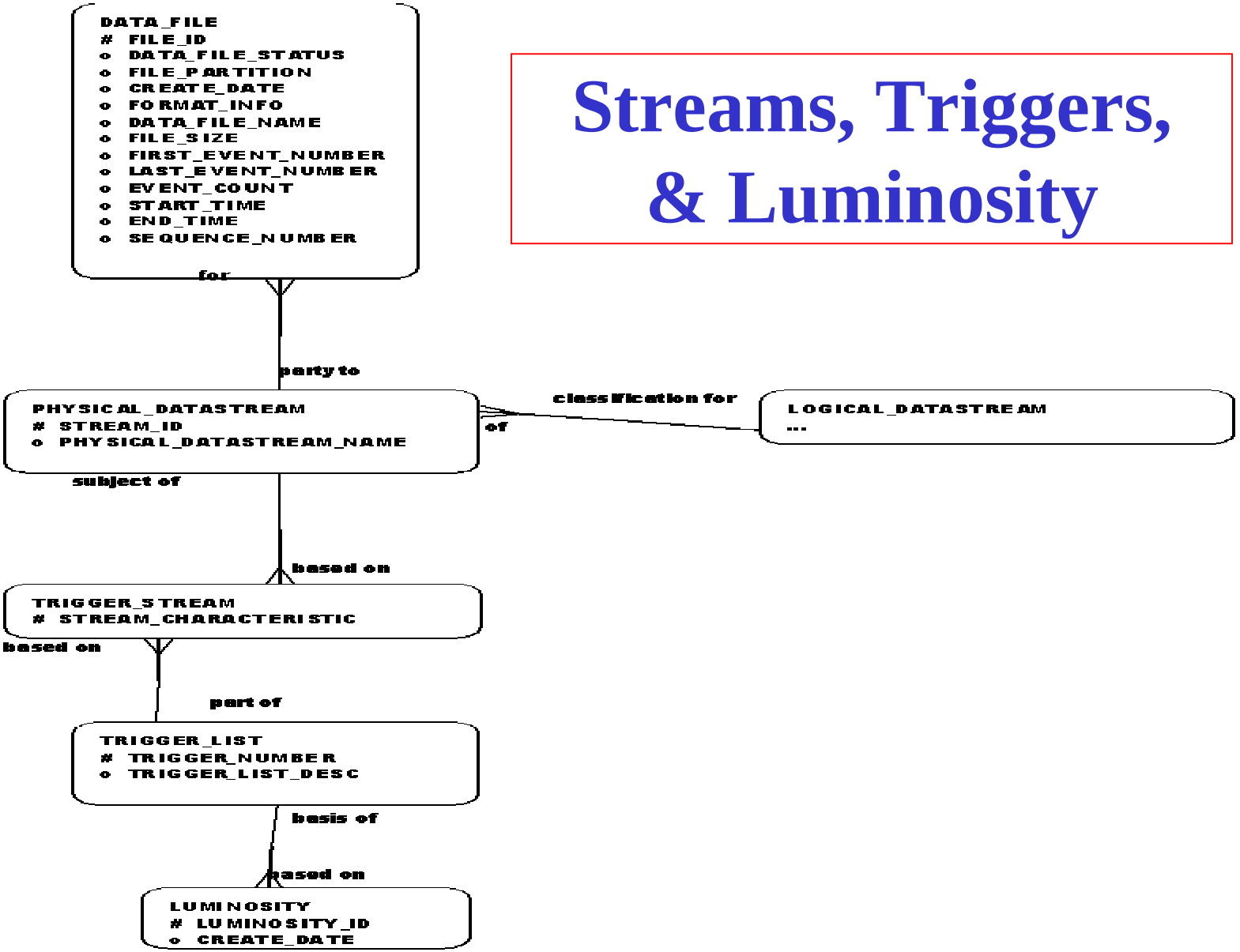
Event & File Catalogs



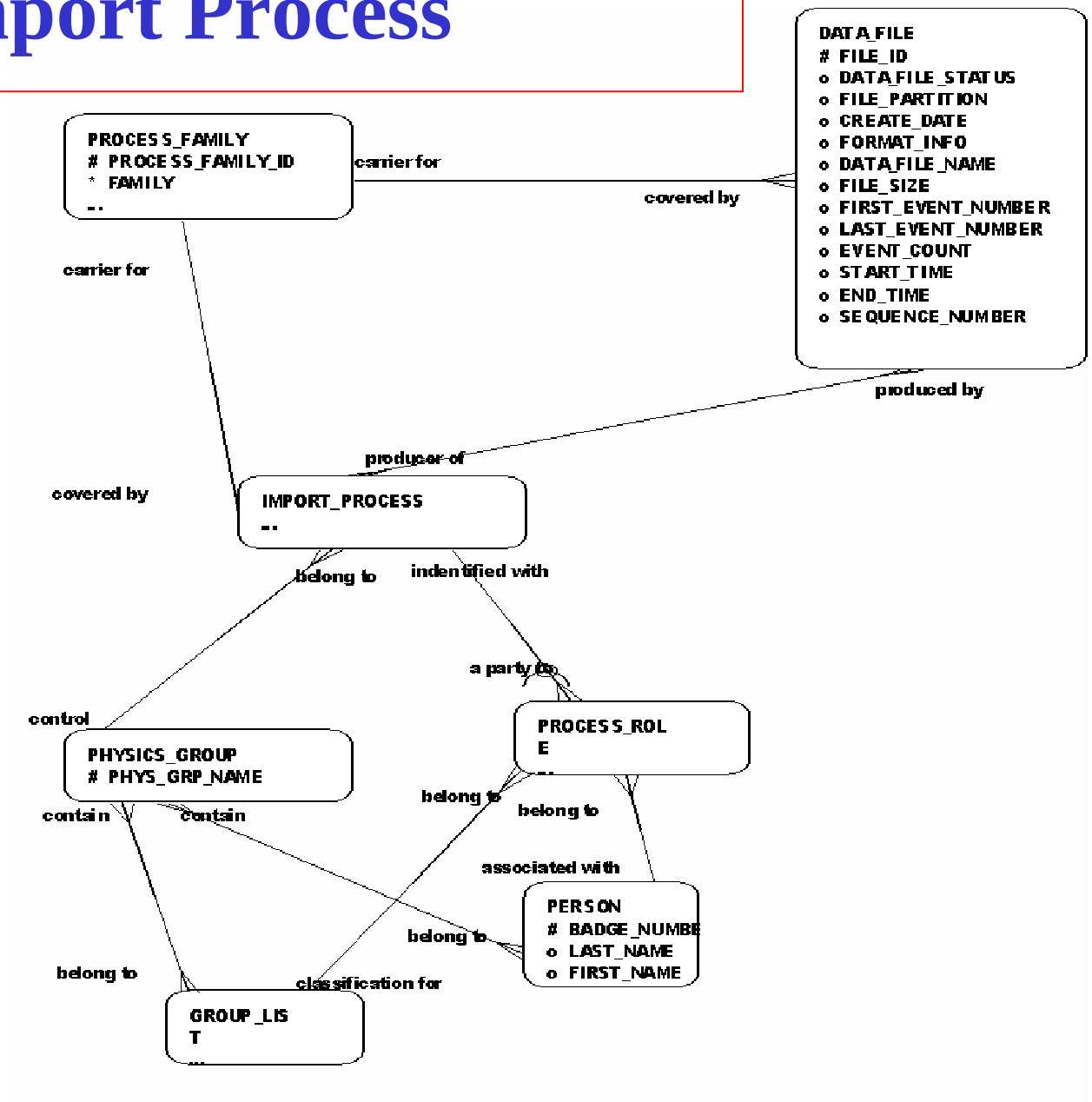
Physical Data Location



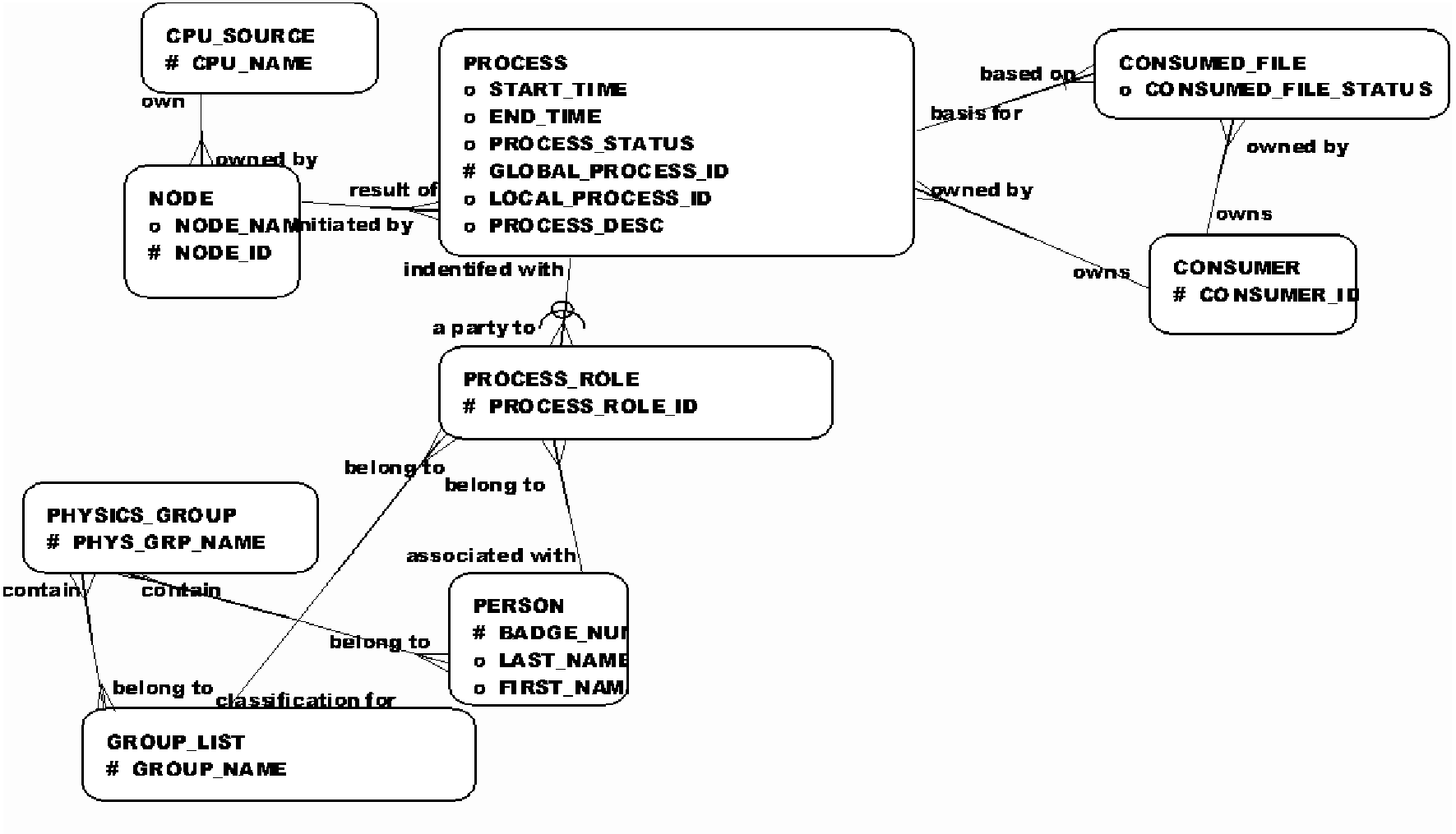
Streams, Triggers, & Luminosity



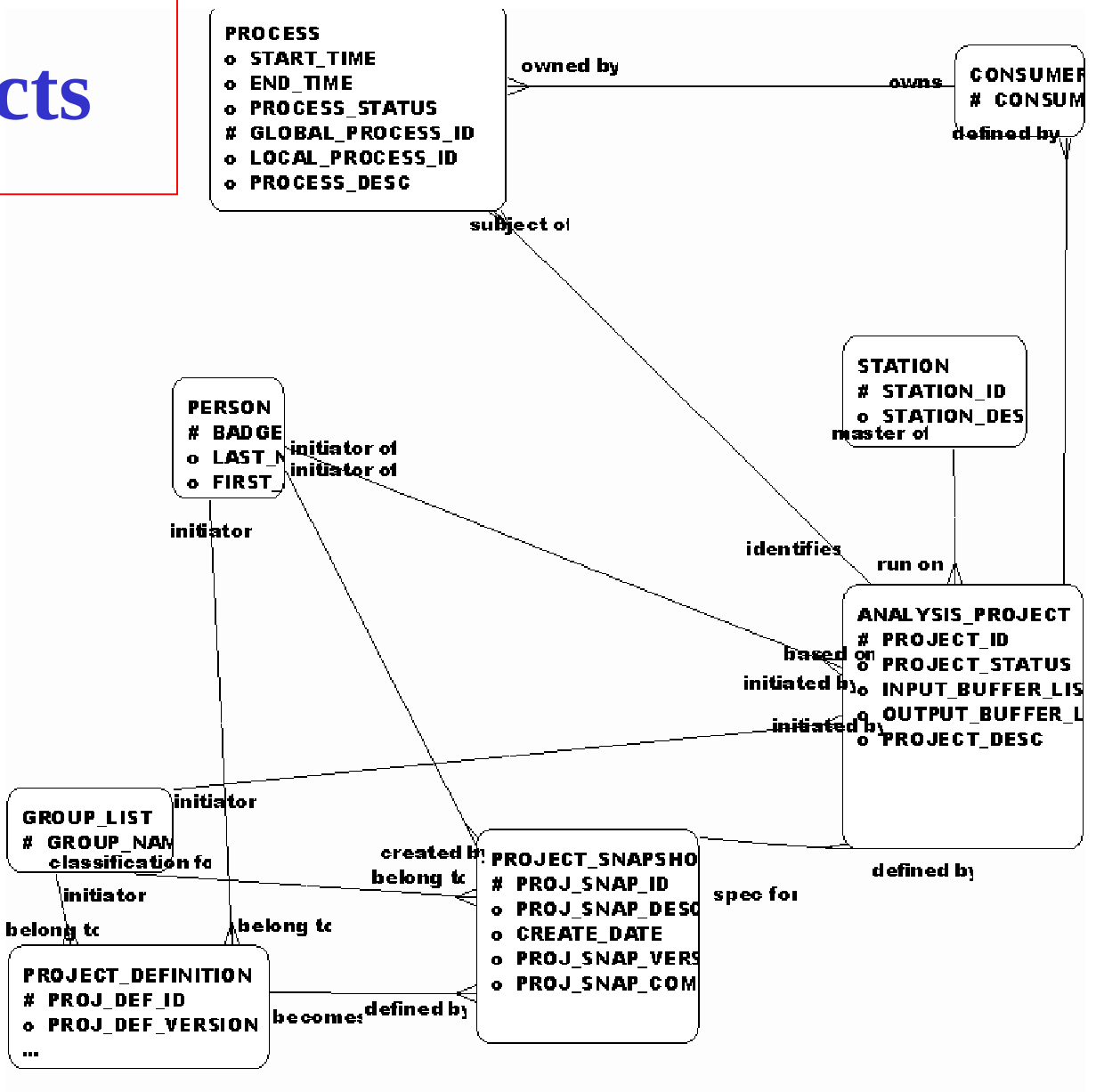
Import Process



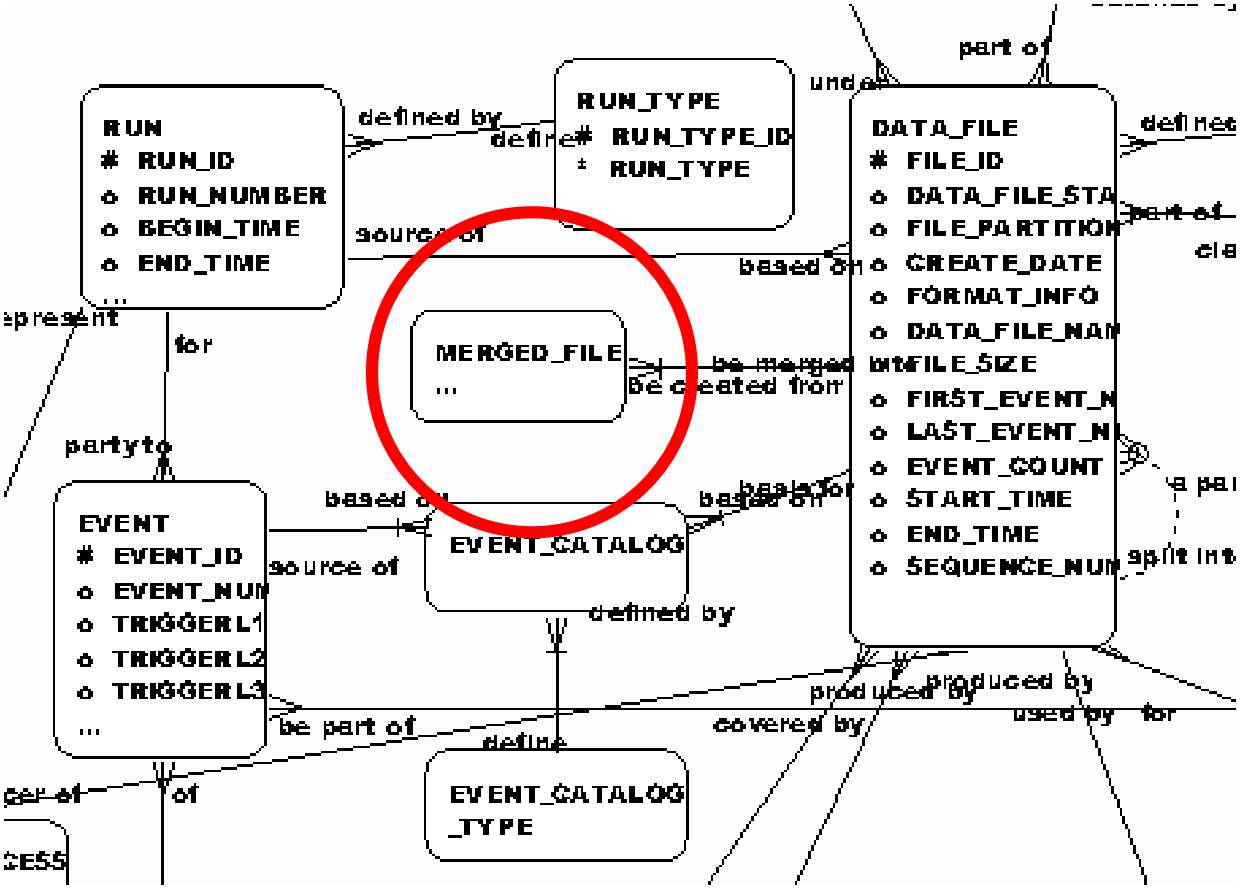
Process & Consumers



Projects


PROCESS
o START_TIME
o END_TIME
o PROCESS_STATUS
GLOBAL_PROCESS_ID
o LOCAL_PROCESS_ID
o PROCESS_DESC
owned by
owns
CONSUMER
CONSUM
defined by
subject of
STATION
STATION_ID
o STATION_DES
master of
PERSON
BADGE
o LAST_N
o FIRST_
initiator of
initiator of
initiator
identifies
run on
based on
initiated by
initiated by
ANALYSIS_PROJECT
PROJECT_ID
o PROJECT_STATUS
o INPUT_BUFFER_LIS
o OUTPUT_BUFFER_L
o PROJECT_DESC
initiator
GROUP_LIST
GROUP_NAM
classification fo
created by
belong to
PROJECT_SNAPSHO
PROJ_SNAP_ID
o PROJ_SNAP_DESC
o CREATE_DATE
o PROJ_SNAP_VERS
o PROJ_SNAP_COM
defined by
spec for
initiator
belong to
belong to
PROJECT_DEFINITION
PROJ_DEF_ID
o PROJ_DEF_VERSION
...
becomes
defined by

Processing Chains:
The Lineage Table



Import and Export

Import & Export

- **Previous examples**
  - Monte Carlo processing chain.
  - Raw data from DAQ.
  - Merging files
- **Motivation for using Python in import scripts**
  - We did NOT want to create our own language, or write yet another parser.
  - Using Python class semantics and inheritance features allows us to write clear, concise descriptions of the import data.
  - This was quick and easy. There may be other scripting languages which are just as good, but...
- **Digging into the Python scripts**

Python Import Scripts

- 8 pages of python to illustrate the 3 examples

MC Data Import

```
SQL> l
  1  select file_id,data_file_name,run_id,file_size from data_files
  2  where exists (select run_id from import_processes
  3  where import_processes.run_id=data_files.run_id)
  4* order by file_id
SQL> /

   FILE_ID DATA_FILE_NAME                RUN_ID  FILE_SIZE
---------- ----------------------------- ---------- ----------
    914201 tjj1001.gen                   100000     204800
    914202 tjjpy1001.raw                  100000     204800
    914203 tjjpy1002.raw                  100000     204800
    914204 tjjpy1003.raw                  100000     102400
    914205 tjjpy1004.raw                  100000     102400
    914206 tjjpy1005.raw                  100000     102400
    914207 tjjpyre1001.250                 100000     102400
    914208 tjjpyre1002.250                 100000     102400
    914209 tjjpyre1003.250                 100000     102400
    914210 tjjpyre1004.250                 100000     102400
    914211 tjjpyre1005.250                 100000     102400
```

RAW Data Import

```
SQL> l
  1  select file_id,data_file_name,run_id,file_size from data_files
  2  where exists (select run_id from import_processes
  3  where import_processes.run_id=data_files.run_id)
  4* order by file_id
SQL> /

   FILE_ID DATA_FILE_NAME                RUN_ID  FILE_SIZE

  914212 jet_0010000_01_01.RAW             10000    1000000
   914213 jet_0010000_02_02.RAW             10000    1048576
   914214 jet_0010000_03_03.RAW             10000    1048576
   914215 jet_0010000_04_04.RAW             10000    1048576
   914216 jet_0010000_05_05.RAW             10000    1048576
   914217 electron_0010000_01_01.RAW          10000    1048576
   914218 electron_0010000_02_02.RAW          10000    1048576
   914219 electron_0010000_03_03.RAW          10000    1048576
   914220 electron_0010000_04_04.RAW          10000    1048576
   914221 electron_0010000_05_05.RAW          10000    1048576
   914222 muon_0010000_01_01.RAW              10000    1048576
   914223 muon_0010000_02_02.RAW              10000    1048576
   914224 muon_0010000_03_03.RAW              10000    1048576
   914225 muon_0010000_04_04.RAW              10000    1048576
   914226 muon_0010000_05_05.RAW              10000    1048576
   914227 electron_jet_0010000_01_01.RAW      10000    1048576
   914228 electron_jet_0010000_02_02.RAW      10000    1048576
   914229 muon_jet_0010000_01_01.RAW          10000    1048576
   914230 electron_muon_0010000_02_02.RAW  10000    1048576
   914231 electron_muon_jet_0010000_01_01.RAW 10000    1048576
```

The Lineage Table

```
SQL> l
  1  select l.file_id_source,l.file_id_dest,
  2  l.parent_sequence,l.source_split,l.dest_merge,f.data_file_name
  3  from file_lineages l, data_Files f
  4  where l.file_id_source=f.file_id
  5* order by file_id_dest
SQL> /

FILE_ID_SOURCE FILE_ID_DEST PARENT_SEQUENCE SOURCE_SPLIT DEST_MERGE DATA_FILE_NAME
-------------- ------------ --------------- ------------ ---------- ------------------------------
        914201       914202               1            1          0 tjj1001.gen
        914201       914203               1            1          0 tjj1001.gen
        914201       914204               1            1          0 tjj1001.gen
        914201       914205               1            1          0 tjj1001.gen
        914201       914206               1            1          0 tjj1001.gen
        914202       914207               1            0          0 tjjpy1001.raw
        914203       914208               1            0          0 tjjpy1002.raw
        914204       914209               1            0          0 tjjpy1003.raw
        914205       914210               1            0          0 tjjpy1004.raw
        914206       914211               1            0          0 tjjpy1005.raw
        914212       914232               1            0          1 jet_0010000_01_01.RAW
        914213       914232               2            0          1 jet_0010000_02_02.RAW
        914215       914232               4            0          1 jet_0010000_04_04.RAW
        914216       914232               5            0          1 jet_0010000_05_05.RAW
        914214       914232               3            0          1 jet_0010000_03_03.RAW
        914220       914233               4            0          1 electron_0010000_04_04.RAW
        914217       914233               1            0          1 electron_0010000_01_01.RAW
        914219       914233               3            0          1 electron_0010000_03_03.RAW
        914218       914233               2            0          1 electron_0010000_02_02.RAW
        914221       914233               5            0          1 electron_0010000_05_05.RAW
        914222       914234               1            0          1 muon_0010000_01_01.RAW
        914223       914234               2            0          1 muon_0010000_02_02.RAW
        914224       914234               3            0          1 muon_0010000_03_03.RAW
        914225       914234               4            0          1 muon_0010000_04_04.RAW
        914226       914234               5            0          1 muon_0010000_05_05.RAW
```

Merged Files

FILE_ID	DATA_FILE_NAME	RUN_ID	FILE_SIZE
914232	jet_0010000_01_05.EDU010	10001	102400
914233	electron_0010000_01_05.EDU010	10001	102400
914234	muon_0010000_01_05.EDU010	10001	102400

Project Snapshot

```
SQL> l
  1  select proj_snap_desc,project_files.file_id,data_file_name,file_number
  2  from project_snapshots,project_files,data_files
  3  where project_snapshots.proj_snap_id=project_files.proj_snap_id
  4  and project_files.file_id=data_files.file_id
  5* and proj_snap_desc='MV-19981113152722'
SQL> /

PROJ_SNAP_DESC          FILE_ID DATA_FILE_NAME              FILE_NUMBER
----------------------- ------- --------------------------- -----------
MV-19981113152722       914202 tjjpy1001.raw                  1
MV-19981113152722       914203 tjjpy1002.raw                  2
MV-19981113152722       914204 tjjpy1003.raw                  3
MV-19981113152722       914205 tjjpy1004.raw                  4
MV-19981113152722       914206 tjjpy1005.raw                  5
MV-19981113152722       914207 tjjpyre1001.250                6
MV-19981113152722       914208 tjjpyre1002.250                7
MV-19981113152722       914209 tjjpyre1003.250                8
MV-19981113152722       914201 tjj1001.gen                   9
MV-19981113152722       914210 tjjpyre1004.250               10
MV-19981113152722       914211 tjjpyre1005.250               11

11 rows selected.
```

Consumer Running Project

```
SQL> l
  1  select proj_snap_desc,project_snapshots.proj_snap_id,
  2  file_number,consumed_files.consumer_id,processes.global_process_id,
  3  consumed_file_status,station_desc from
  4  project_snapshots,processes,consumed_files,analysis_projects,
  5  stations
  6  where project_snapshots.proj_snap_id=
  7  analysis_projects.proj_snap_id and
  8  analysis_projects.project_id=
  9  processes.project_id
 10   and stations.station_id=analysis_projects.station_id
 11   and exists(select proj_snap_id from consumed_files
 12* where consumed_files.proj_snap_id=project_snapshots.proj_snap_id)
SQL> /
```

PROJ_SNAP_DESC	PROJ_SNAP_ID	FILE_NUMBER	CONSUMER_ID	GLOBAL_PROCESS_ID	CONSUMED_FILE_STATUS	STATION_DESCRIPTION
top_jet_test	12	2	532	328	CONSUMED	STATION5
top_jet_test	12	1	532	328	CONSUMED	STATION5

Information Services

- start of performance details --- will refine and use with Grau robot testing
- parses and analyses the log file of all significant events in the system
  - see Info Server charts and statistics

Station Master

- **Barely implemented in Prototype**
- **Starts Projects - hands out part of Station Disk Cache to a Project Master**
- **Receives filled parts of Disk Cache back from a Project Master**
- **Synchronizes disk cache inventory with DB (SM startup & termination of a project)**
- **Locks file in disk cache if in use by a Project (local knowledge)**
- **Authorizes and arranges Intra Station and Inter Station copying of files disk to disk**

Project Master and Stagers (encp clients)

- **Project Master in control of**
  - **giving file names to Analysis Programs & keeping their scorecard**
  - **the staging processes and the use of disk buffers assigned to the Project (gives them work to do)**
  - **communications with Global Optimizer**
  - **recording information on Analysis Programs (Consumers) and Files consumed/created - in DB**
  - **handling errors - rejoins, modes of access etc.**
- **C++ code - multiple threads + CORBA threads**
  - **Have had one project master on a Linux machine handle up to 60 consumers running on the same m/c - ran out of process slots at some point > 60**

Stagers

- Little more than just a call to Enstore's encp program
- Receives status block back from encp at end of a file transfer - records info.
- No intelligence in this component - all smarts are in the Project Master code
- No reason a Stager could not be interfaced to any reasonable staging or tape reading system

Resource Management

- Optimizing the use of all resources simultaneously is not possible
- Depending on which is the most scarce resource different strategies must be employed for Resource Management
- SAM assumes that the Tape Robot/tape drives/connecting network will be a scarce global resource which requires management
- SAM assumes a part of that management is done by Enstore

Resource Management

- If Tape to Disk I/O is the bottleneck then Stagers must be configured carefully to be running in parallel, getting all the CPU they need, whenever they need it, with the total number of TCP/IP pipes well matched to the max delivery possible and to the disk write I/o bandwidth
  - SAM Station must be configurable to arrange this ...
- If CPU is the bottleneck a batch system must meter the jobs
- If reading particular disk I/o bandwidth is the bottleneck then some replication of data must be considered - or more disk controllers and disk must be purchased
  - people must figure this out by looking at statistics and performance
- If uncoordinated access to large datasets creates a thrashing of files on disk and from robot and means analysis jobs are frequently re-fetching files from robot - then more coordination and scheduling must be imposed
  - SAM can provide statistics to help understand this

Resource Management

- The bottom line is that no analysis job can analyze a file which is not yet resident on disk
- If it can analyze data faster than the delivery system can deliver it and ship out the resulting files (if necessary) then the job is I/O bound and should be made eligible to be slotted in and out by some batch system each time it gets hung up waiting for data, (or will just hang using no cpu until data arrives).
- If it takes much longer to analyze than the time to fetch the data then our interest is only in global fair-shares allocation of I/o resources so that it is kept fed with data in accordance with overall global optimization of delivery of data by access mode & project type.

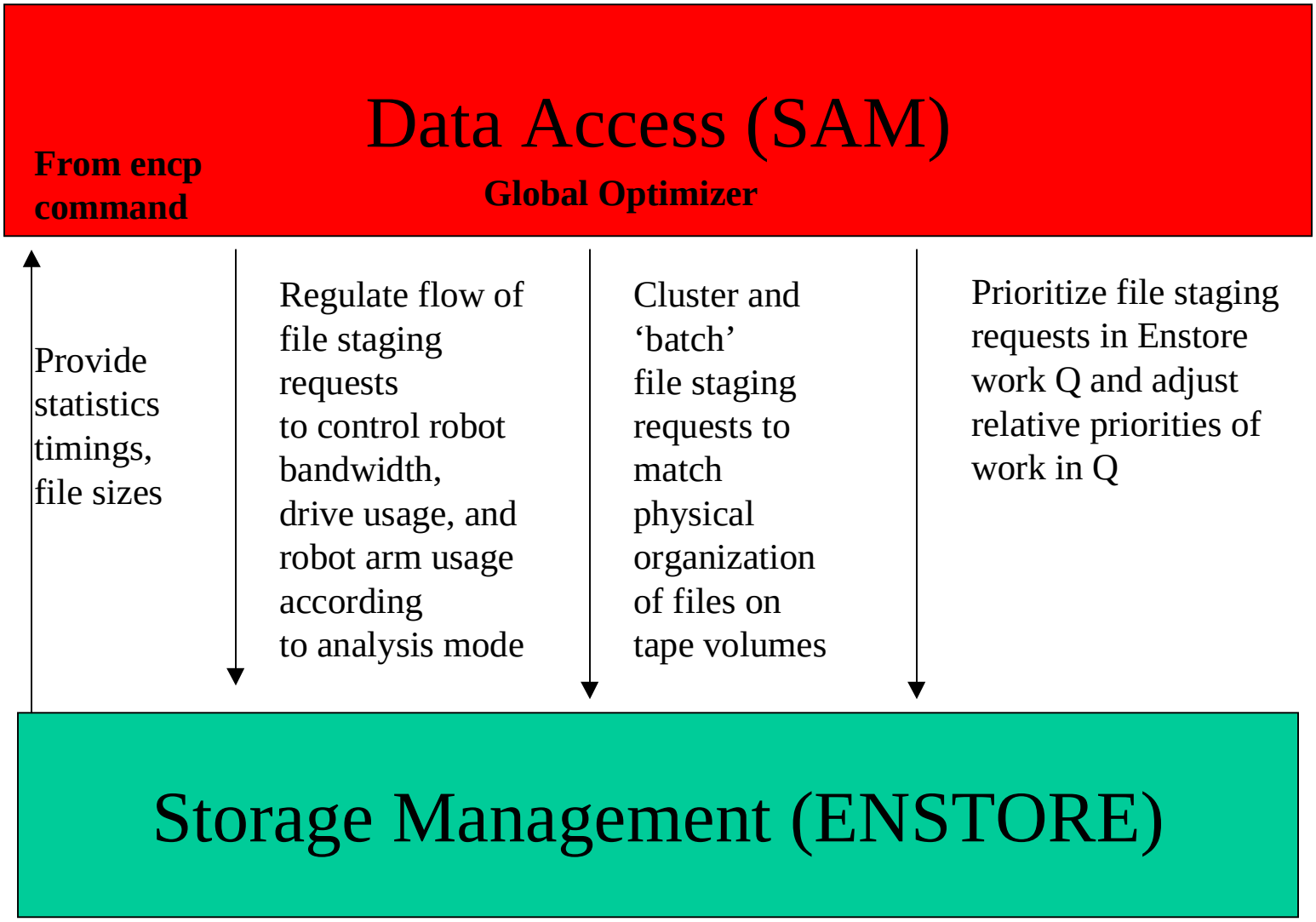
Resource Management

- Static or semi-static configurable **Station Resources**
  - who can use station for what purposes
  - allocations of managed disk on a particular station
  - policies for disk cache flushing.
  - policies for amount and nature of intra-station and inter-station movement of data
- **I/O** (tape to disk) - largely handled by SAM layer (with potential interplay with Batch system if used)
- **cpu** and disk read I/o - largely handled by Batch system (with potential interplay with SAM layer)

What a Batch System and/or Farm Production Layer might expect of SAM

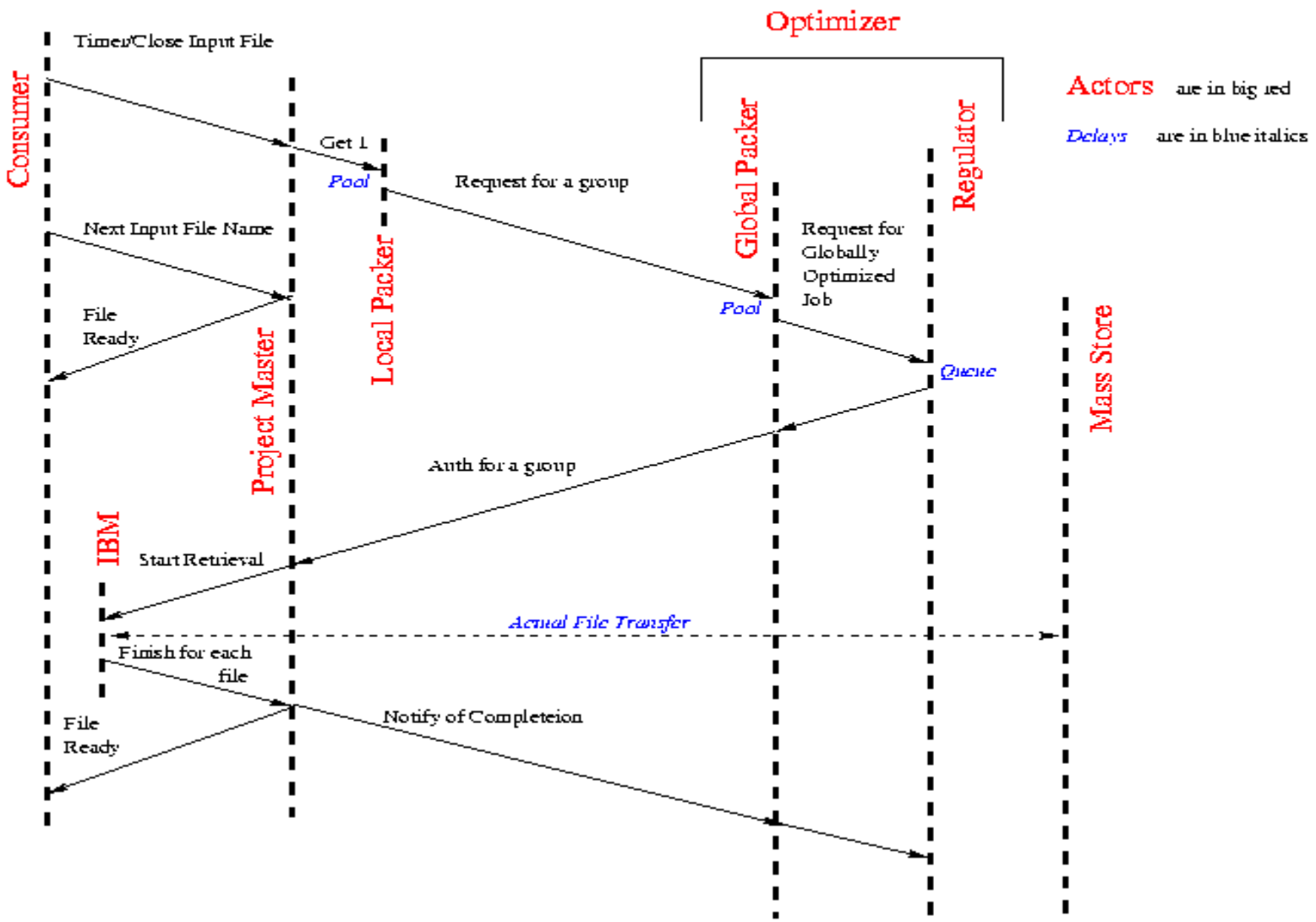
- the Data Access Layer (SAM) of the data handling system will deliver data to users without benefit of a batch system to control resources or start jobs.
- The Data Access Layer's job is not to globally optimize all CPU, disk I/o, scratch space resources - but to arrange for data to be specified, delivered to disk, cached on disk.

Resource Management Interfaces

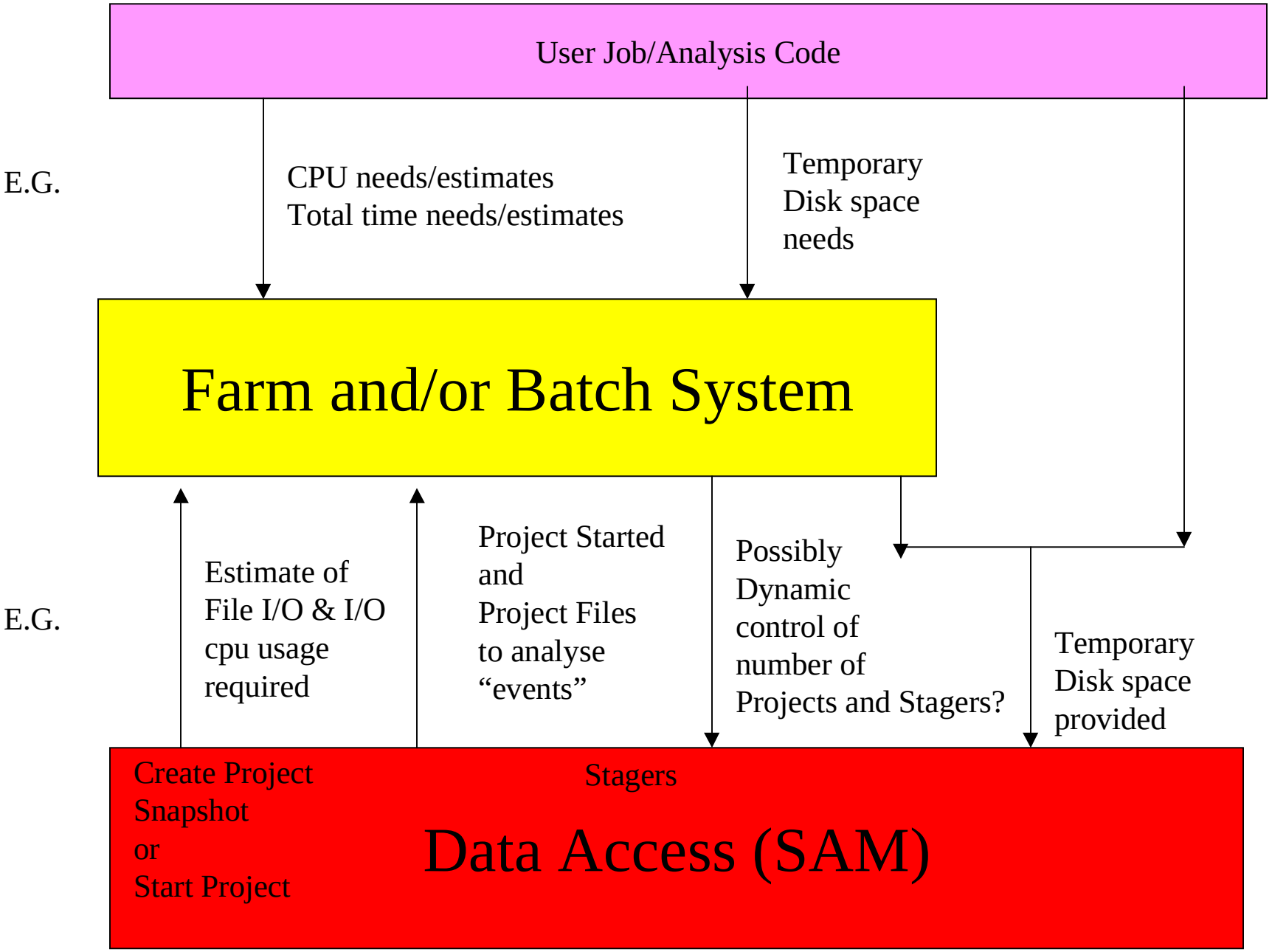


Reorders work to traverse tape in physical file order
Implements Priority and Aging of work and re-orders work Q appropriately

Event Trace for Reading Production Files from the Mass Store

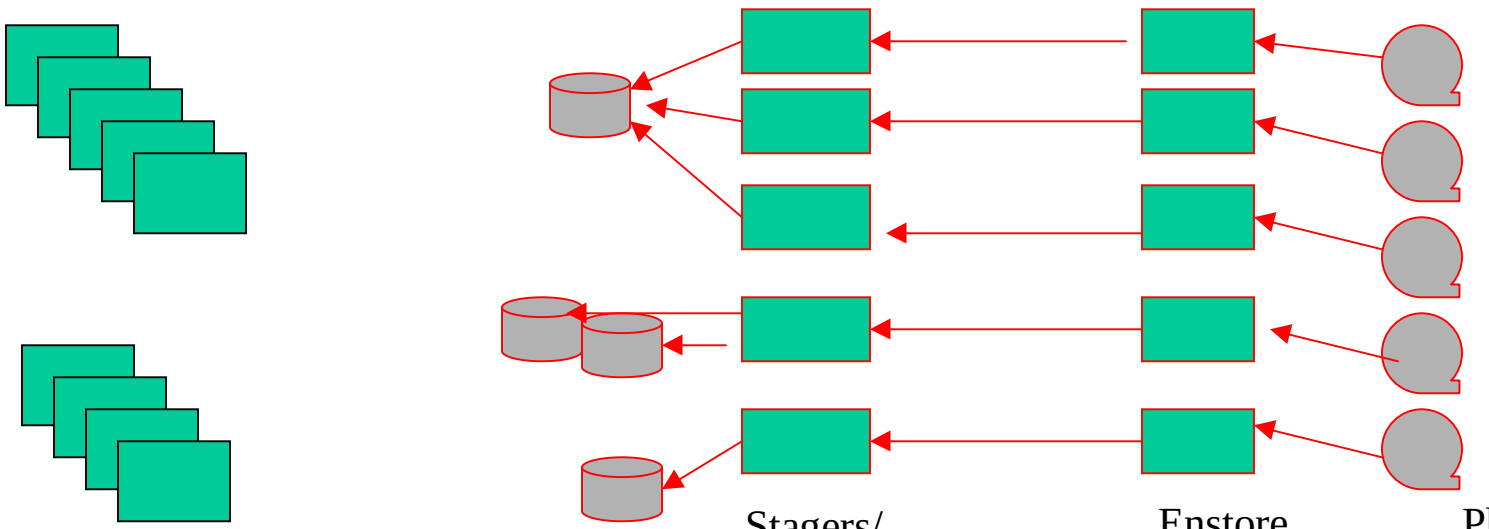


Resource Management Interfaces (TBD)



I/O (tape to disk) resources

- A SAM station must be able to be configured so that N encp staging jobs are run on a particular station hardware --- because N is the number of such TCP/IP pipes which is matched to the network hardware and to the max tape I/o bandwidth from Enstore which is expected to flow into that system.



A Note on the Batch System - Time to pay attention to this

- SAM has continually said that it expects a Batch System to be used in many places where SAM is used - wants to design interface with Batch System (Later) - but does not rely on a batch system.
- Last Farms minutes state -- “Not determined yet, but CDF is leaning toward using LSF as it's batch system”
- Is this a Joint Project? Autonomous CDF decision? Subject to design of interfaces with data handling? What about the Job Scheduler piece of Platform Computing’s suite? Cost? No of licences?

Documentation and Use Cases

- http://runiicomputing.fnal.gov/sam

finis