Pipeline Back-End

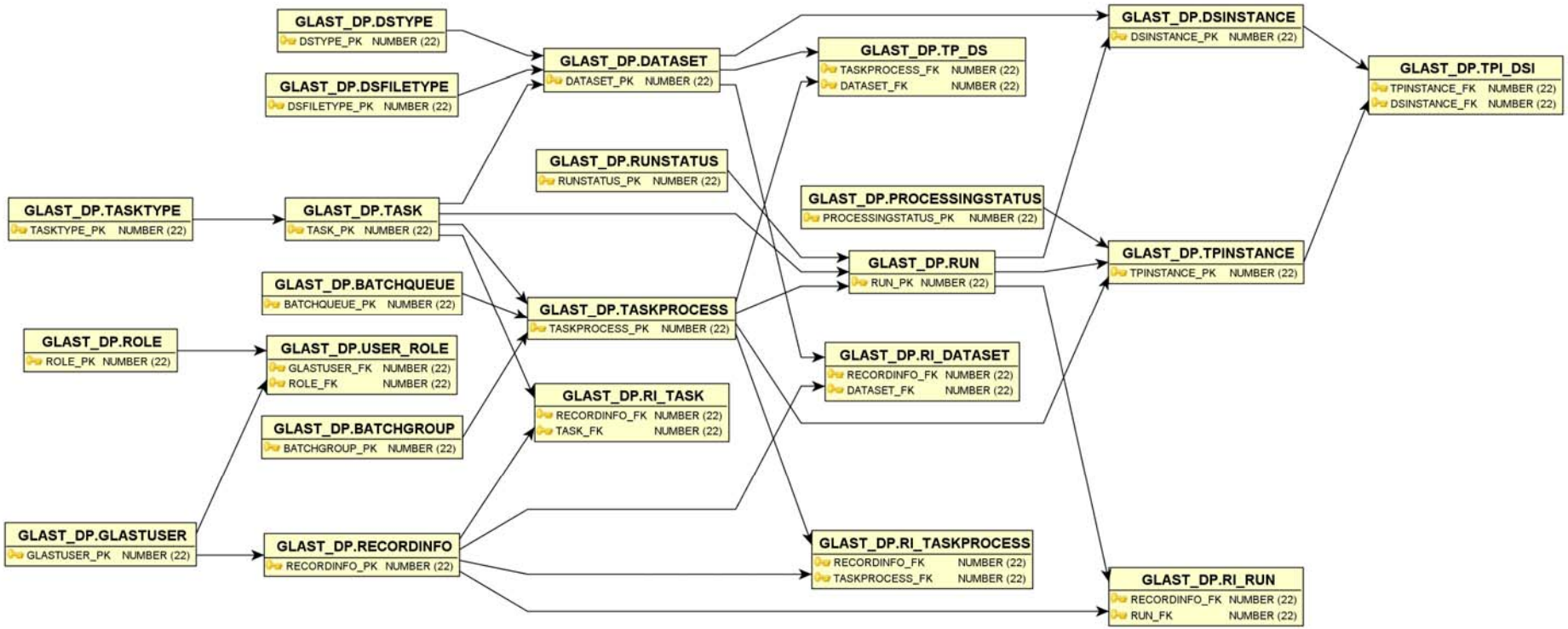
Daniel Flath

GLAST Offline Software
Developers' Workshop
March, 2005

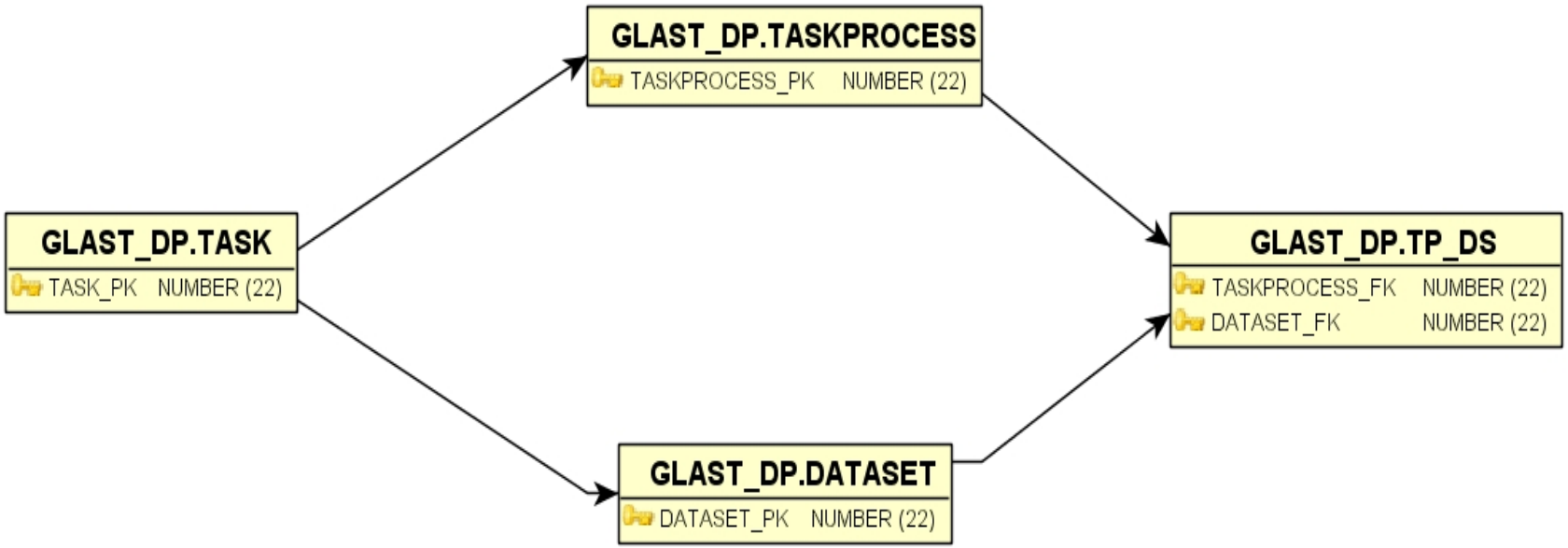
PDB Overview

- PDB organized into 2 groups of tables:
  - User Config
  - Instantiation
- PDB built of 3 Table Types:
  - Regular (Task, Run, etc.)
  - Enumerated (TaskType, RunStatus, etc.)
  - Relational (TP_DS, TPI_DSI, etc.)

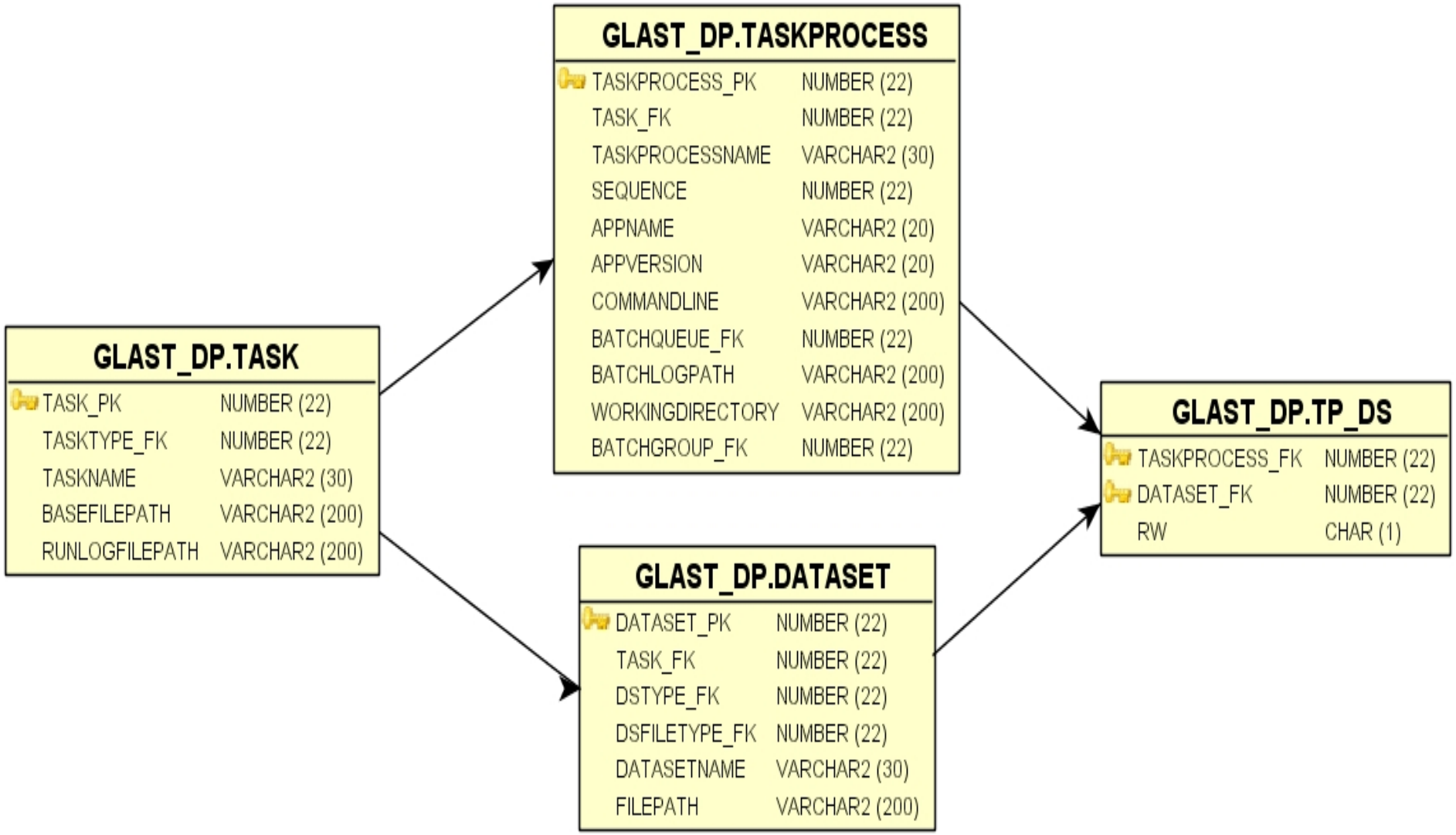
Database Schema



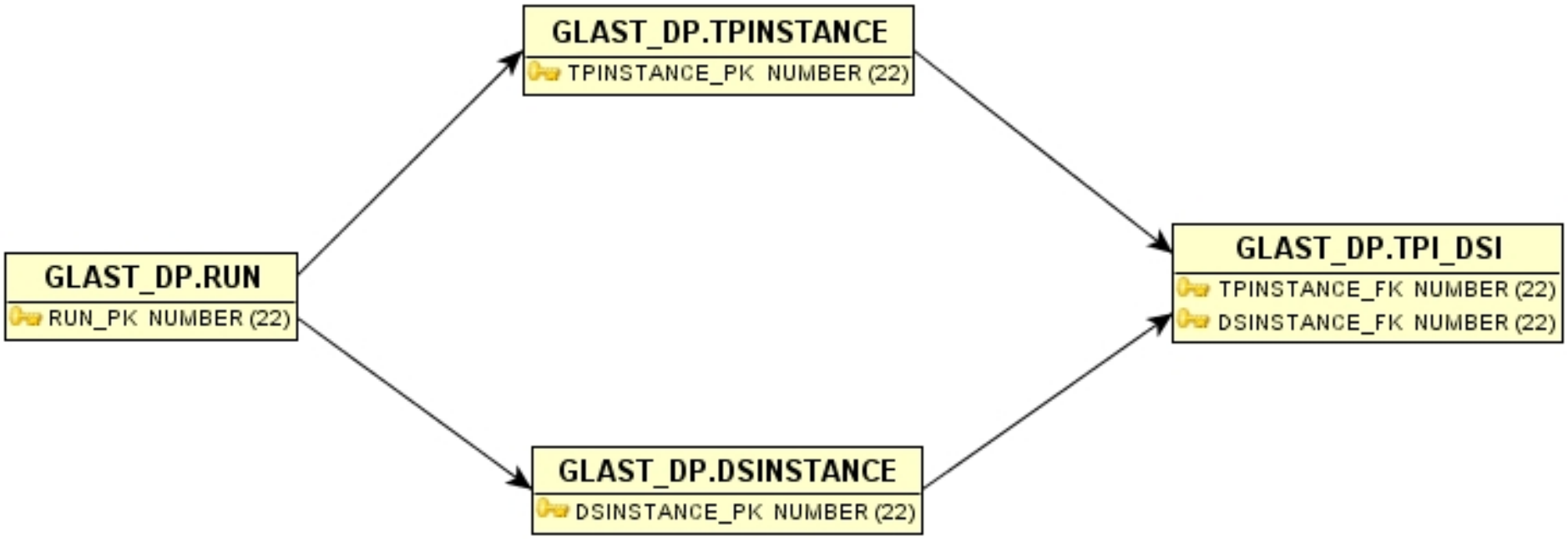
User Config Overview



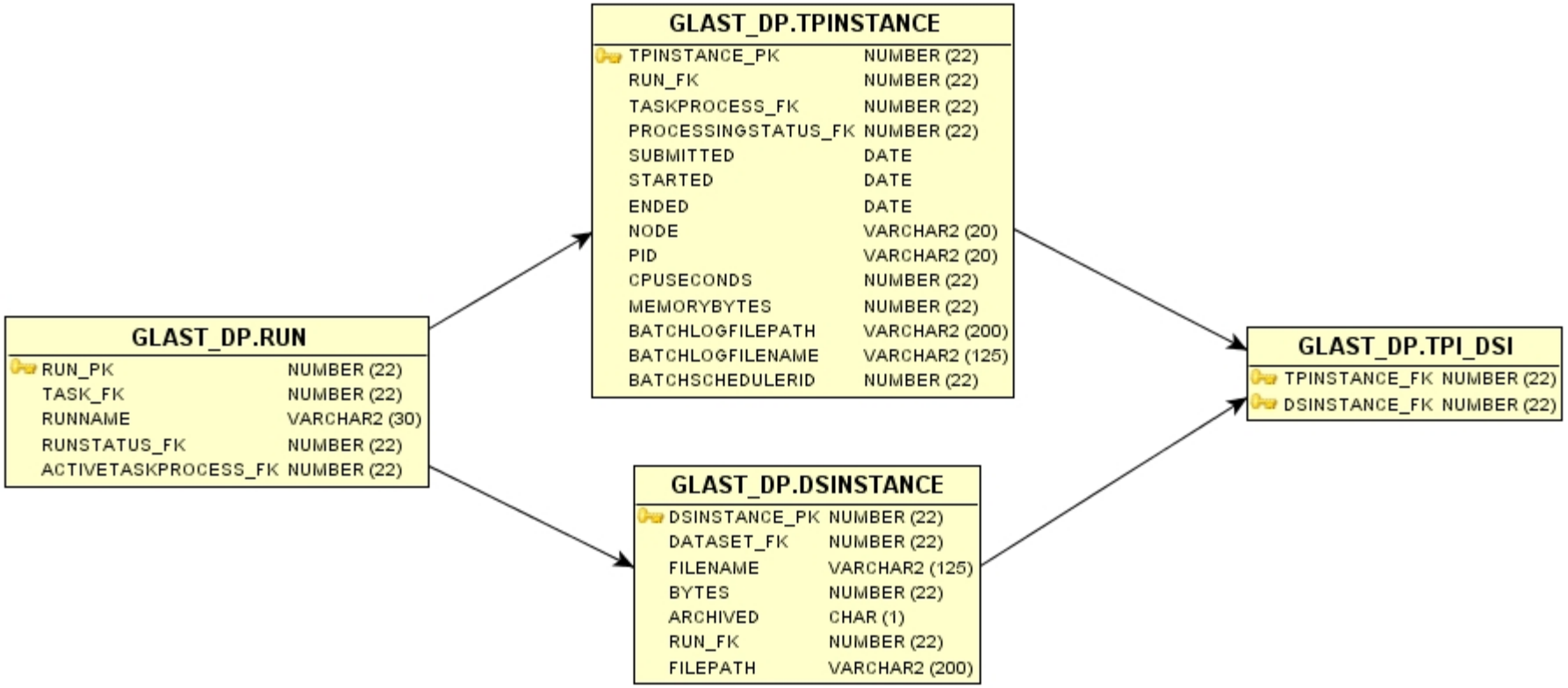
User Config Detail



Instantiation Overview



Instantiation Detail



DSInstance

- Instantiation of a ‘Dataset’ record
- Represents a file on disk
- Tracks a data file’s relationship to processing (attached to TPInstance records via TPI_DSI records)
- Tracks a data file’s location on disk and/or tape when file moved or archived

TPInstance

- Represents a processing step in a task
- A State Machine in a sense
- As a TPI evolves through possible values of ProcessingStatus it's state determines status of the Run that owns it
- These relationships are given below
- Currently ProcessingStatus of a TPI and the RunStatus of it's owner (infrequently) get out of synch
  - Need to fix this – Working on this currently while incorporating Navid's more robust LSF interface. Requires some modifications to SP's to make TPI & Run flag updates atomic

Run & Proc Status

- Possible combinations:

RunStatus	**ProcStatus**
WAITING	(no TPI record yet)
RUNNING	PREPARED
RUNNING	SUBMITTED
RUNNING	RUNNING
FINALIZING	FINALIZING
DONE	END_SUCCESS
FAILED	END_FAILURE

Scheduler

- State Machine – gathers information about current state of active Runs (➔TPIs)
  - Gathers Runs in ‘Finalizing’ state, checks corresponding LSF log-files and promotes Run/TPI status accordingly
  - Gathers Runs in ‘Wait’ state
    - Calculates next TP’s input and target DSes
    - Submits next TP to LSF, passing DS info to user’s wrapper script
    - This TP is monitored by a waiting process on the DPF server
      - At the moment this causes a DB connection to persist during processing

Scheduler++

- With incorporation of Navid's LSF interface we will see these changes:
  - Scheduler will only launch runs
  - TPI & Run status flags (and associated time-stamps) will be updated passively by a callBack script invoked by Navid
  - Persistent DB Connections will disappear

Task Dependencies

- Can now use DSes from a Task in a subsequently executed Task
- User must launch this dependant explicitly from a preceding Task
  - Think this might be done automatically by adding a relational table associating TPs (or DSes) with Tasks.