Run History DB and MetaCat

- Data discovery

Ana Paula Vizcaya Hernández

Norm Buchanan

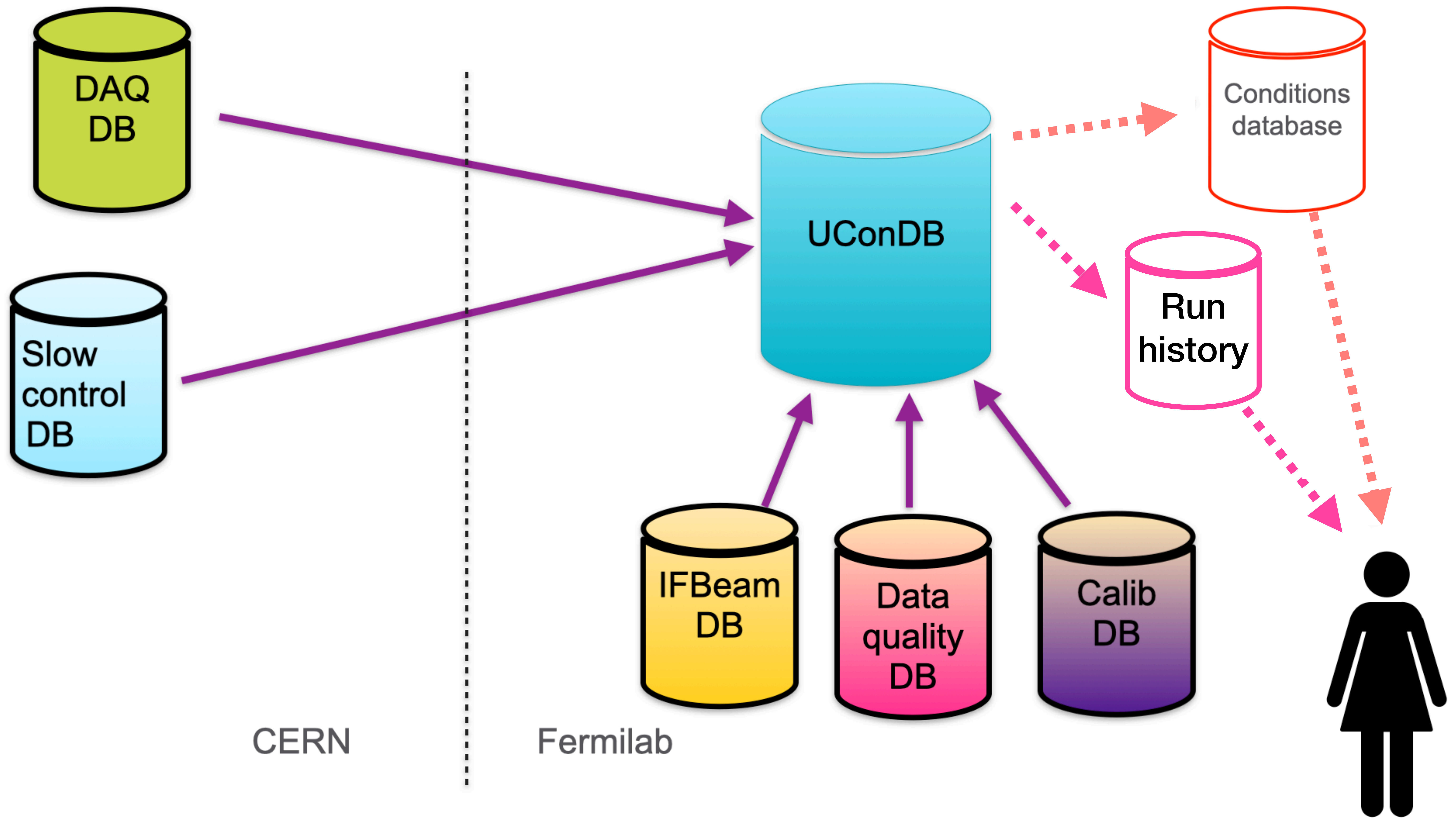
14/3/2023

Colorado State University

Run History DB / Conditions DB

Run History DB

- It is a relational DB with just a selection of run conditions parameters
- Purpose: get runs/files for runs with specific configurations.
- Goal: integrate the Run Conditions DB with MetaCat



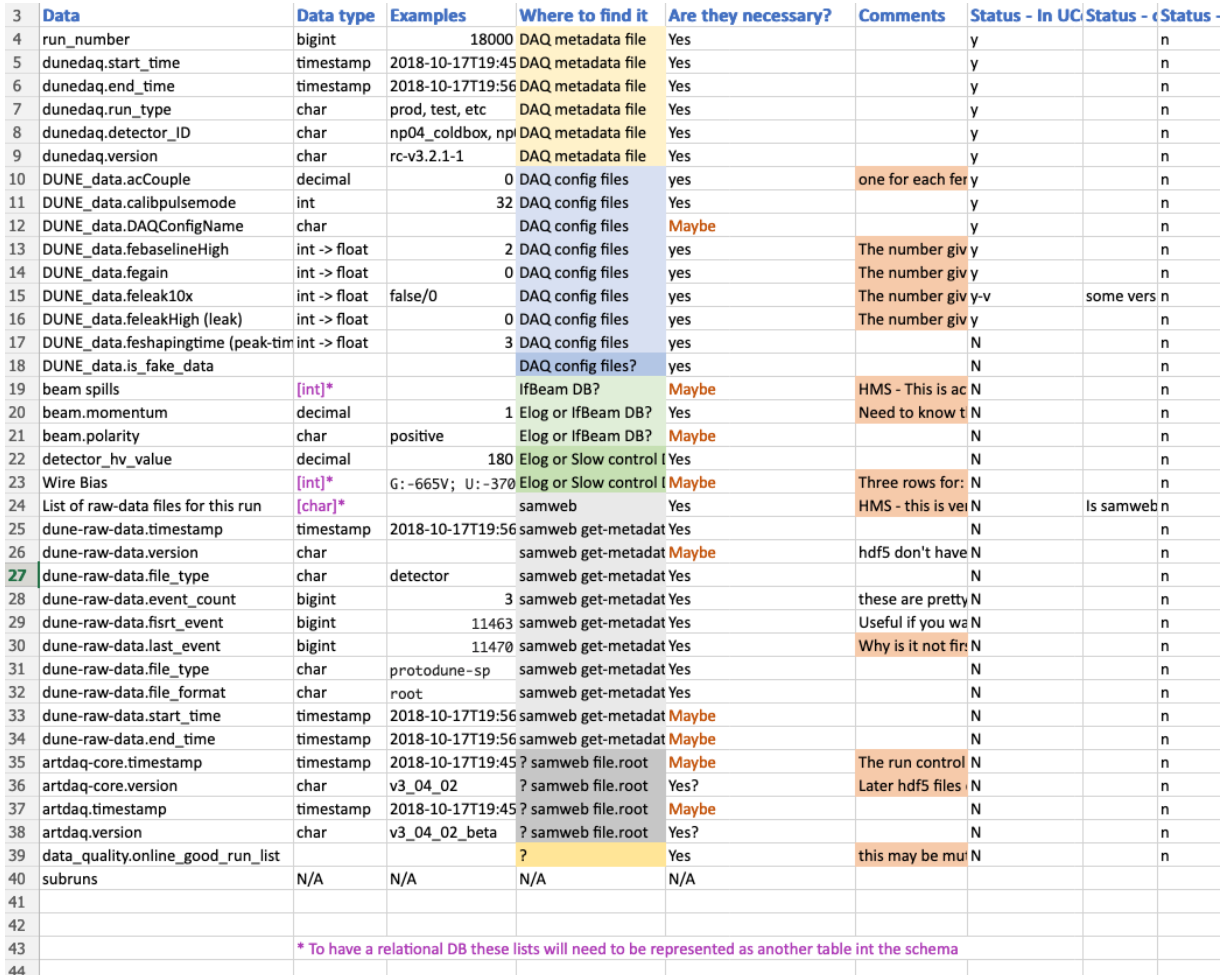
Understanding the data

What data do we want to store?

- Heidi showed a list of data coming from a ProtoDUNE run 1 file
- Data used during ProtoDUNE run 1

I created a spreadsheet with all the data:

https://colostate-my.sharepoint.com/:x:/g/personal/avizcaya_colostate_edu/EeUJJ4LBS_RBnIPqQVHjag4BbvnwaNdHIDoEfJ1Qbfl7cQ?e=nuwngB



3	Data	Data type	Examples	Where to find it	Are they necessary?	Comments	Status - In UC	Status -	Status -
4	run_number	bigint	18000	DAQ metadata file	Yes		y		n
5	dunedaq.start_time	timestamp	2018-10-17T19:45	DAQ metadata file	Yes		y		n
6	dunedaq.end_time	timestamp	2018-10-17T19:56	DAQ metadata file	Yes		y		n
7	dunedaq.run_type	char	prod, test, etc	DAQ metadata file	Yes		y		n
8	dunedaq.detector_ID	char	np04_coldbox, np	DAQ metadata file	Yes		y		n
9	dunedaq.version	char	rc-v3.2.1-1	DAQ metadata file	Yes		y		n
10	DUNE_data.acCouple	decimal	0	DAQ config files	yes	one for each fe	y		n
11	DUNE_data.calibpulsemode	int	32	DAQ config files	Yes		y		n
12	DUNE_data.DAQConfigName	char		DAQ config files	Maybe		y		n
13	DUNE_data.febaselineHigh	int -> float	2	DAQ config files	yes	The number giv	y		n
14	DUNE_data.fegain	int -> float	0	DAQ config files	yes	The number giv	y		n
15	DUNE_data.feleak10x	int -> float	false/0	DAQ config files	yes	The number giv	y-v	some vers	n
16	DUNE_data.feleakHigh (leak)	int -> float	0	DAQ config files	yes	The number giv	y		n
17	DUNE_data.feshapingtime (peak-tim	int -> float	3	DAQ config files	yes		N		n
18	DUNE_data.is_fake_data			DAQ config files?	yes		N		n
19	beam spills	[int]*		IfBeam DB?	Maybe	HMS - This is ac	N		n
20	beam.momentum	decimal	1	Elog or IfBeam DB?	Yes	Need to know t	N		n
21	beam.polarity	char	positive	Elog or IfBeam DB?	Maybe		N		n
22	detector_hv_value	decimal	180	Elog or Slow control	Yes		N		n
23	Wire Bias	[int]*	G:-665V; U:-370	Elog or Slow control	Maybe	Three rows for:	N		n
24	List of raw-data files for this run	[char]*		samweb	Yes	HMS - this is ve	N	Is samweb	n
25	dune-raw-data.timestamp	timestamp	2018-10-17T19:56	samweb get-metadat	Yes		N		n
26	dune-raw-data.version	char		samweb get-metadat	Maybe	hdf5 don't have	N		n
27	dune-raw-data.file_type	char	detector	samweb get-metadat	Yes		N		n
28	dune-raw-data.event_count	bigint	3	samweb get-metadat	Yes	these are pretty	N		n
29	dune-raw-data.fisrt_event	bigint	11463	samweb get-metadat	Yes	Useful if you wa	N		n
30	dune-raw-data.last_event	bigint	11470	samweb get-metadat	Yes	Why is it not fir	N		n
31	dune-raw-data.file_type	char	protodune-sp	samweb get-metadat	Yes		N		n
32	dune-raw-data.file_format	char	root	samweb get-metadat	Yes		N		n
33	dune-raw-data.start_time	timestamp	2018-10-17T19:56	samweb get-metadat	Maybe		N		n
34	dune-raw-data.end_time	timestamp	2018-10-17T19:56	samweb get-metadat	Maybe		N		n
35	artdaq-core.timestamp	timestamp	2018-10-17T19:45	? samweb file.root	Maybe	The run control	N		n
36	artdaq-core.version	char	v3_04_02	? samweb file.root	Yes?	Later hdf5 files	N		n
37	artdaq.timestamp	timestamp	2018-10-17T19:45	? samweb file.root	Maybe		N		n
38	artdaq.version	char	v3_04_02_beta	? samweb file.root	Yes?		N		n
39	data_quality.online_good_run_list			?	Yes	this may be mu	N		n
40	subruns	N/A	N/A	N/A	N/A				
41									
42									
43		* To have a relational DB these lists will need to be represented as another table int the schema							
44									

Understanding the data

Versioning

What can change?

- The value of one entry of one run
- All the values of one column
- Add a new column
- It needs its own table

Raw files

More than one per run

- To have a relational DB we need to have one value per row
- Several parameters depend on the raw file, like: event number, first event, etc
- They change from .root to .hdf5, so their properties change
- It needs its own table

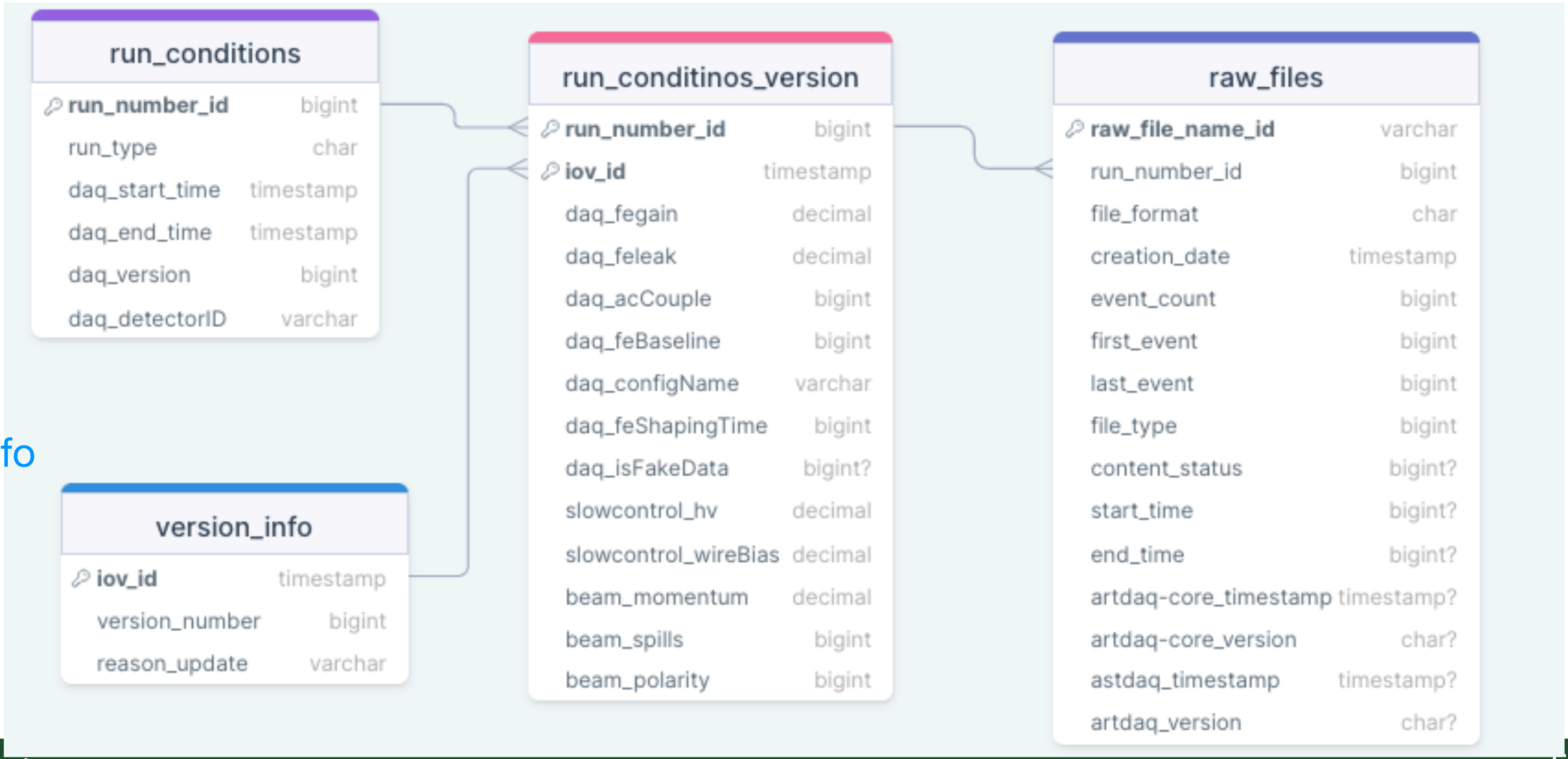
Proposed schema following normalization rules

Static table

Changing table

One value per row or create another table

Versions info



run_conditions	
run_number_id	bigint
run_type	char
daq_start_time	timestamp
daq_end_time	timestamp
daq_version	bigint
daq_detectorID	varchar

run_conditinos_version	
run_number_id	bigint
iov_id	timestamp
daq_fegain	decimal
daq_feleak	decimal
daq_acCouple	bigint
daq_feBaseline	bigint
daq_configName	varchar
daq_feShapingTime	bigint
daq_isFakeData	bigint?
slowcontrol_hv	decimal
slowcontrol_wireBias	decimal
beam_momentum	decimal
beam_spills	bigint
beam_polarity	bigint

raw_files	
raw_file_name_id	varchar
run_number_id	bigint
file_format	char
creation_date	timestamp
event_count	bigint
first_event	bigint
last_event	bigint
file_type	bigint
content_status	bigint?
start_time	bigint?
end_time	bigint?
artdaq-core_timestamp	timestamp?
artdaq-core_version	char?
astdaq_timestamp	timestamp?
artdaq_version	char?

version_info	
iov_id	timestamp
version_number	bigint
reason_update	varchar

How to implement it

Two options

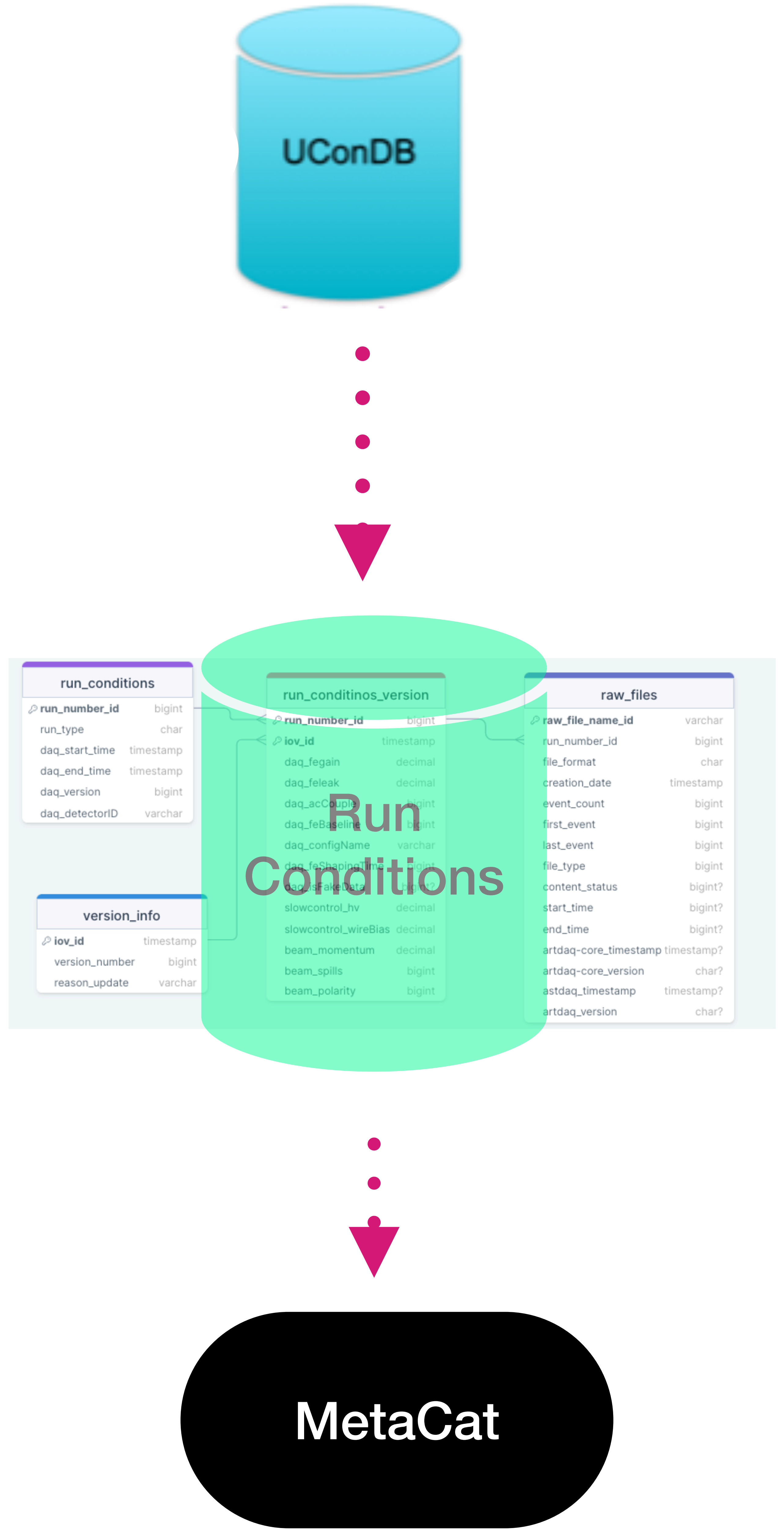
1. Use our own postgreSQL DB at FNAL
2. Use FNAL Conditions DB (different from ProtoDUNE conditions DB)

Our own postgreSQL DB

Use the schema as is.

Steps

1. Create schema at new FNAL DB pdunehd_prod
2. Write code to fill the DB with all the required data
3. Write a python API to comunicate with MetaCat



run_conditions	
run_number_id	bigint
run_type	char
daq_start_time	timestamp
daq_end_time	timestamp
daq_version	bigint
daq_detectorID	varchar

version_info	
iov_id	timestamp
version_number	bigint
reason_update	varchar

run_conditinos_version	
run_number_id	bigint
iov_id	timestamp
daq_fegain	decimal
daq_feleak	decimal
daq_acCouple	bigint
daq_feBaseline	bigint
daq_configName	varchar
daq_feShapingTime	bigint
daq_isFakeData	bigint?
slowcontrol_hv	decimal
slowcontrol_wireBias	decimal
beam_momentum	decimal
beam_spills	bigint
beam_polarity	bigint

raw_files	
raw_file_name_id	varchar
run_number_id	bigint
file_format	char
creation_date	timestamp
event_count	bigint
first_event	bigint
last_event	bigint
file_type	bigint
content_status	bigint?
start_time	bigint?
end_time	bigint?
artdaq-core_timestamp	timestamp?
artdaq-core_version	char?
astdaq_timestamp	timestamp?
artdaq_version	char?

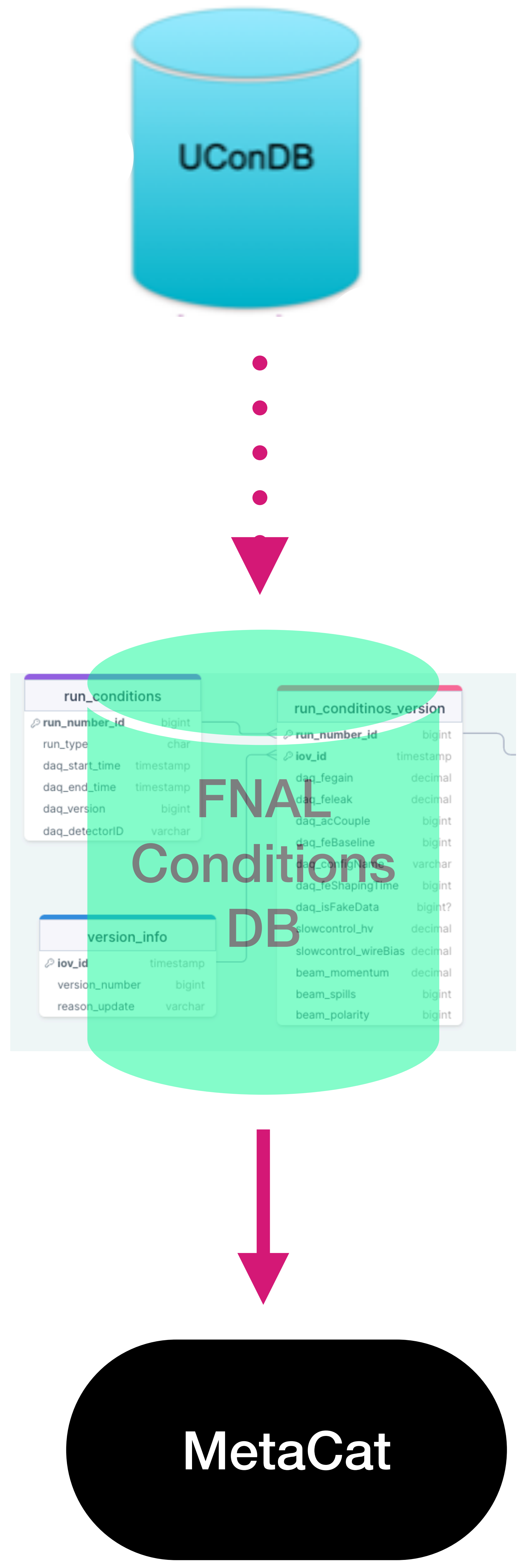
FNAL Conditions DB

Use modified schema where the raw-file info is directly stored in MetaCat and we just have one conditions changing table.

Steps

1. Write code to fill the FNAL Conditions DB with all the required data. There is an API that facilitates inserting data.
2. How to add the raw file extra info?

There already exists a python API to communicate with MetaCat



Which one is better for our use case?

Summary and outlook

- Studied the data
  - where to find it?
  - is it needed?
- Created a proposed schema which handles versioning and raw-data files
- Studied two possibilities of where to put the Run Condition DB
  - FNAL Conditions DB
  - Our own postgreSQL DB

- Choose a Run Conditions DB
- Access all info and send it to UconDB
- Populate the new Run Conditions DB

Thank you

Colorado State University

Backup slides

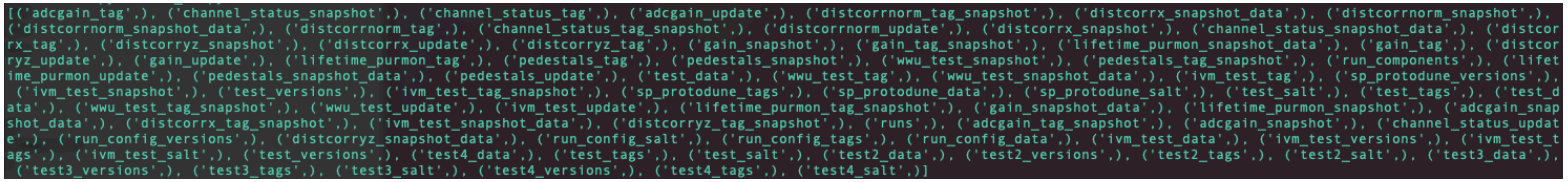
Run History during ProtoDUNE-SP

Data

- It contains 19 parameters
- All of them come from DAQ data
- Part of the pdunesp_prod database and it's located at the folder pdunesp.runs

Questions

- Will the pdunesp_prod db be used for protodune-HD?
- Or should we just create a new folder in this DB



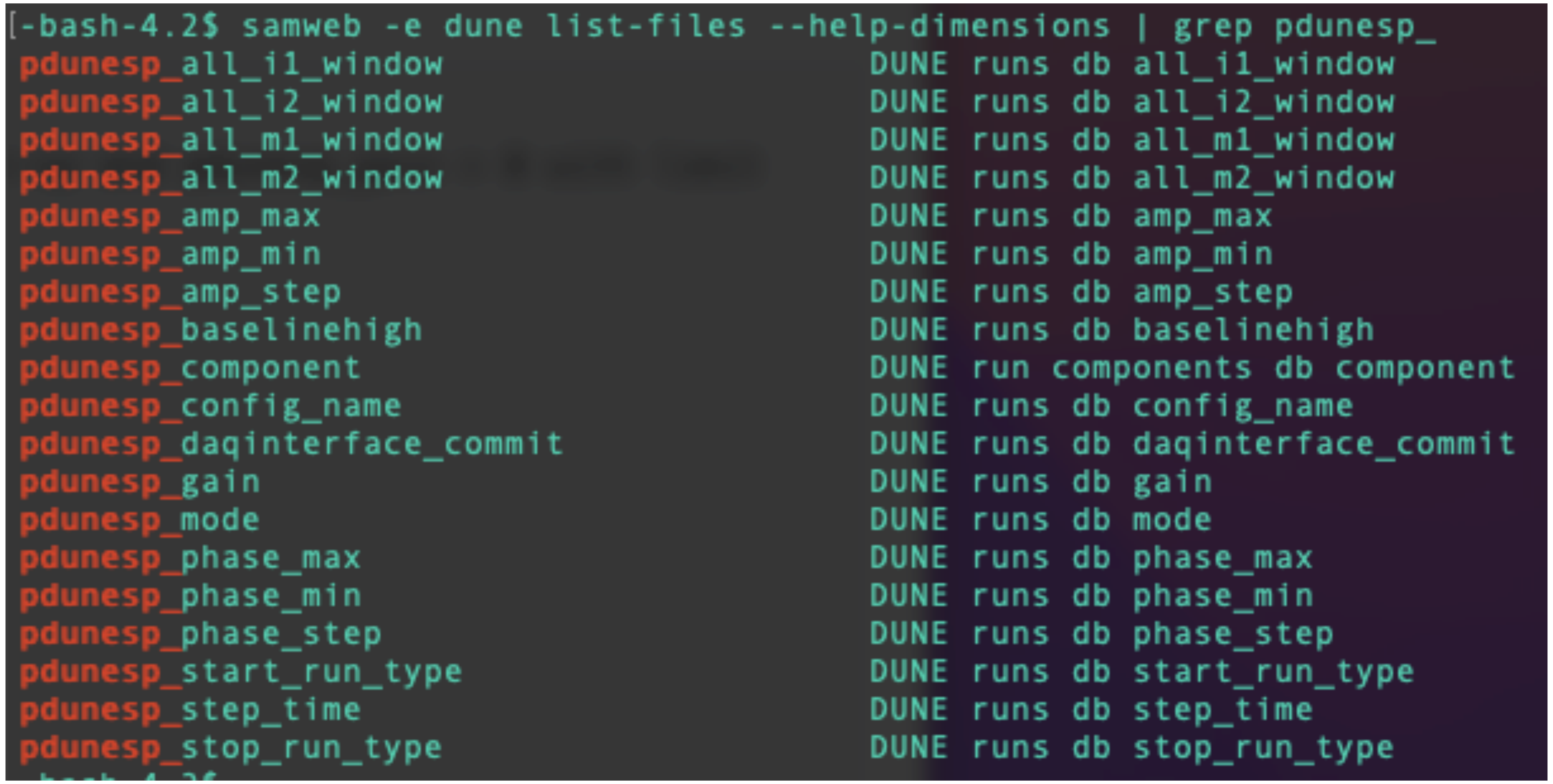
```
[('adcgain_tag',), ('channel_status_snapshot',), ('channel_status_tag',), ('adcgain_update',), ('distcorrnorm_tag_snapshot',), ('distcorrx_snapshot_data',), ('distcorrnorm_snapshot',),
('distcorrnorm_snapshot_data',), ('distcorrnorm_tag',), ('channel_status_tag_snapshot',), ('distcorrnorm_update',), ('distcorrx_snapshot',), ('channel_status_snapshot_data',), ('distcor
rx_tag',), ('distcorryz_snapshot',), ('distcorrx_update',), ('distcorryz_tag',), ('gain_snapshot',), ('gain_tag_snapshot',), ('lifetime_purmon_snapshot_data',), ('gain_tag',), ('distcor
ryz_update',), ('gain_update',), ('lifetime_purmon_tag',), ('pedestals_tag',), ('pedestals_snapshot',), ('wwu_test_snapshot',), ('pedestals_tag_snapshot',), ('run_components',), ('lifet
ime_purmon_update',), ('pedestals_snapshot_data',), ('pedestals_update',), ('test_data',), ('wwu_test_tag',), ('wwu_test_snapshot_data',), ('ivm_test_tag',), ('sp_protodune_versions',),
 ('ivm_test_snapshot',), ('test_versions',), ('ivm_test_tag_snapshot',), ('sp_protodune_tags',), ('sp_protodune_data',), ('sp_protodune_salt',), ('test_salt',), ('test_tags',), ('test_d
ata',), ('wwu_test_tag_snapshot',), ('wwu_test_update',), ('ivm_test_update',), ('lifetime_purmon_tag_snapshot',), ('gain_snapshot_data',), ('lifetime_purmon_snapshot',), ('adcgain_snap
shot_data',), ('distcorrx_tag_snapshot',), ('ivm_test_snapshot_data',), ('distcorryz_tag_snapshot',), ('runs',), ('adcgain_tag_snapshot',), ('adcgain_snapshot',), ('channel_status_updat
e',), ('run_config_versions',), ('distcorryz_snapshot_data',), ('run_config_salt',), ('run_config_tags',), ('run_config_data',), ('ivm_test_data',), ('ivm_test_versions',), ('ivm_test_t
ags',), ('ivm_test_salt',), ('test_versions',), ('test4_data',), ('test_tags',), ('test_salt',), ('test2_data',), ('test2_versions',), ('test2_tags',), ('test2_salt',), ('test3_data',),
 ('test3_versions',), ('test3_tags',), ('test3_salt',), ('test4_versions',), ('test4_tags',), ('test4_salt',)]
```

Parameters

all_i1_window
all_i2_window
all_m1_window
all_m2_window
amp_max
amp_min
amp_step
baselinehigh
component
config_name
daqinterface_commit
gain
mode
phase_max
phase_min
phase_step
start_run_type
step_time
stop_run_type

Run History and SAM

It is possible to query the DB for run selection using SAM.



```
[-bash-4.2$ samweb -e dune list-files --help-dimensions | grep pdunesp_
pdunesp_all_i1_window                     DUNE runs db all_i1_window
pdunesp_all_i2_window                     DUNE runs db all_i2_window
pdunesp_all_m1_window                     DUNE runs db all_m1_window
pdunesp_all_m2_window                     DUNE runs db all_m2_window
pdunesp_amp_max                           DUNE runs db amp_max
pdunesp_amp_min                           DUNE runs db amp_min
pdunesp_amp_step                          DUNE runs db amp_step
pdunesp_baselinehigh                      DUNE runs db baselinehigh
pdunesp_component                         DUNE run components db component
pdunesp_config_name                       DUNE runs db config_name
pdunesp_daqinterface_commit               DUNE runs db daqinterface_commit
pdunesp_gain                              DUNE runs db gain
pdunesp_mode                              DUNE runs db mode
pdunesp_phase_max                         DUNE runs db phase_max
pdunesp_phase_min                         DUNE runs db phase_min
pdunesp_phase_step                        DUNE runs db phase_step
pdunesp_start_run_type                    DUNE runs db start_run_type
pdunesp_step_time                         DUNE runs db step_time
pdunesp_stop_run_type                     DUNE runs db stop_run_type
```

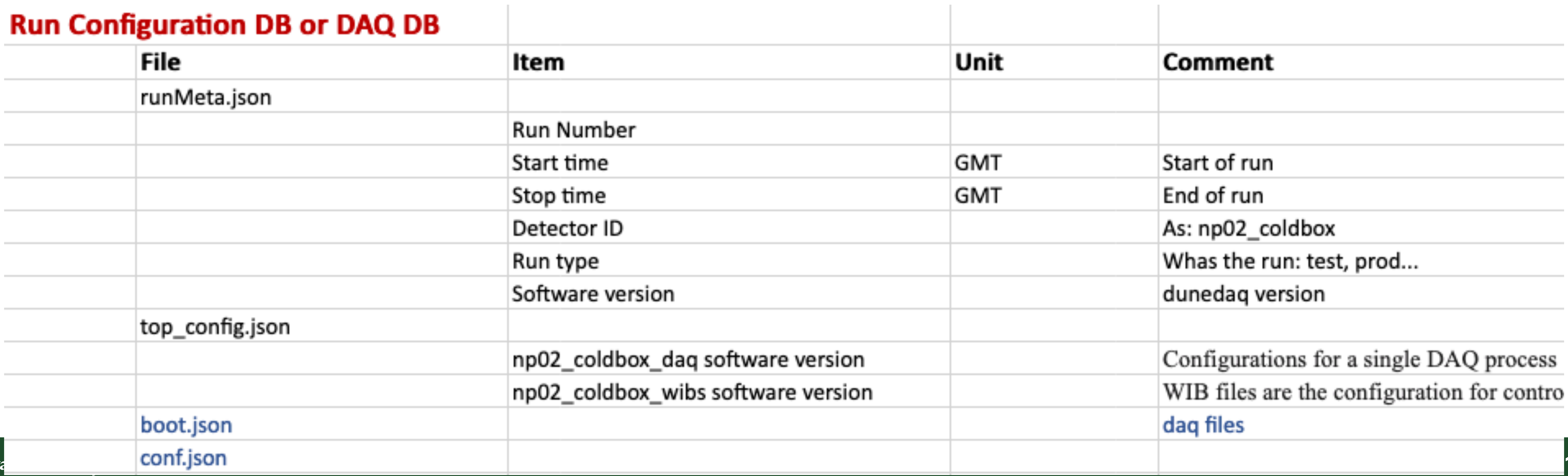
Goal: Make the protoDUNE-HD database accessible via SAM/metacat

Run History DB outlook

- Make a new runs-test folder
- Start with a short list of parameters from the DAQ DB to demonstrate that all the infrastructure is working for data challenge
- Decide on a new folder or a new database
- Make a more complete selection of data
- Work with SAM team to include this database into their infrastructure

Table with possible data for run history

- Created an excel spreadsheet with data that goes to the UConDB
- From there we can select a subset of the data to send to the Run History
- I have included data from the following databases: DAQ, IFBeam and Calibration
- Slow control parameters will be included by Lino
- Table link: https://colostate-my.sharepoint.com/:x:/r/personal/avizcaya_colostate_edu/_layouts/15/Doc.aspx?



Run Configuration DB or DAQ DB

	File	Item	Unit	Comment
	runMeta.json			
		Run Number		
		Start time	GMT	Start of run
		Stop time	GMT	End of run
		Detector ID		As: np02_coldbox
		Run type		Whas the run: test, prod...
		Software version		dunedaq version
	top_config.json			
		np02_coldbox_daq software version		Configurations for a single DAQ process
		np02_coldbox_wibs software version		WIB files are the configuration for contro
	boot.json			daq files
	conf.json			

Raw files for each run

How to treat?

- There can be a lot, 700 files?
- They have several attributes (# events, last and first event, versions…)
- They change from .root to .hdf5, so their properties change
  - Treat this with versioning

Calibration DBs data

Data

- It contains 22 parameters
- Divided in 4 databases
- The correction DBs give a run number

Question

- The data will be transferred to the UConDB, Should we include them in the run history DB? since the databases are not automatically filled this may take some time

Database	Parameter
Electron lifetime	lifetime_TPCC
	lifetime_TPCL
	lifetime_TPCH
	Timestamp
dQ/dx YZ correction	channel
	Run Number
	y
	dy
	z
	dz
	corr
	corr_err
dQ/dx X correction	channel
	Run Number
	x
	dx
	shape
	shape_err
dQ/dx normalization correction	channel
	Run Number
	norm
	norm_err

Adding IFBeam data

Status

- Created executable program (instead of ART module) to transfer data from the IFBeam DB to the UConDB
- Big thanks to Marc Menguel, who is back from extended leave

To - do

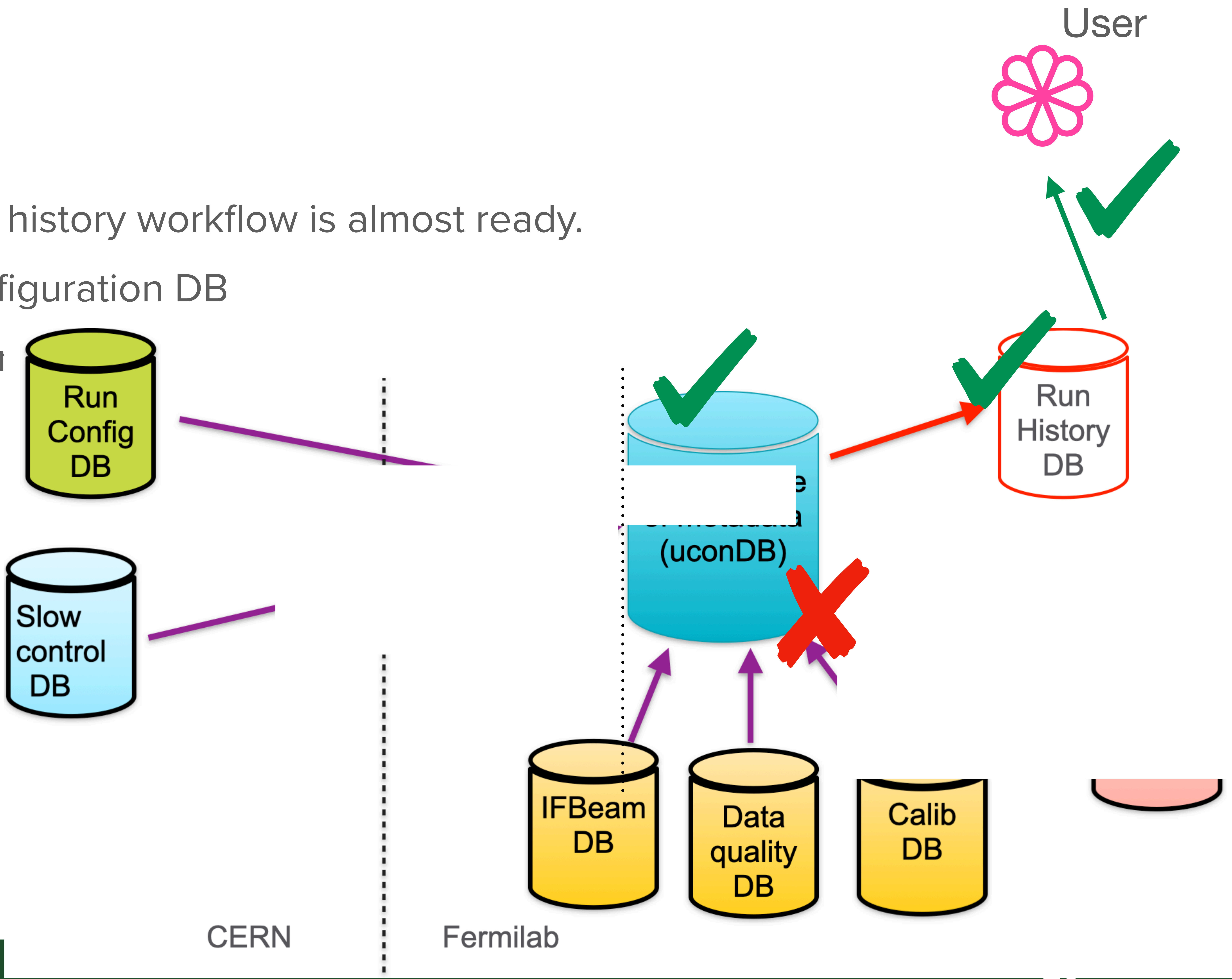
- For the data challenge, add function to transfer data from the IFbeam table in the UConDB to the run history
  - Using the devices 35 devices from Beam Event analysis
  - Take mean/std of each run
  - Suggestions?

Conclusion

- The basic infrastructure of the run history workflow is almost ready.
  - Contains data from the run configuration DB
  - Using a test table in the new rur

To - do

- Add function to transfer data from
- Web interface for the run history?



Run history DB for HD

Location

- It will be a table in the database: pdunehd_prod
- Special permission is needed to access the Db (just for DB experts! - no users)
  - Request access to the DB, create a ticket, and Olga will probably handle it
- Host = ifdbprod.fnal.gov, Port: 5451, and dbname=pdunehd_prod

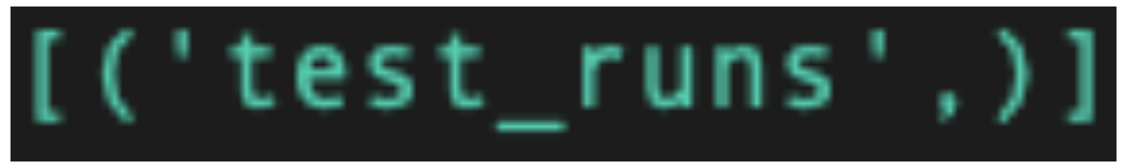
Amenities

- Development database: pdunehd_dev
  - Used for testing
  - Host = ifdb07.fnal.gov, Port: 5448

New Run History DB for protoDUNE-HD

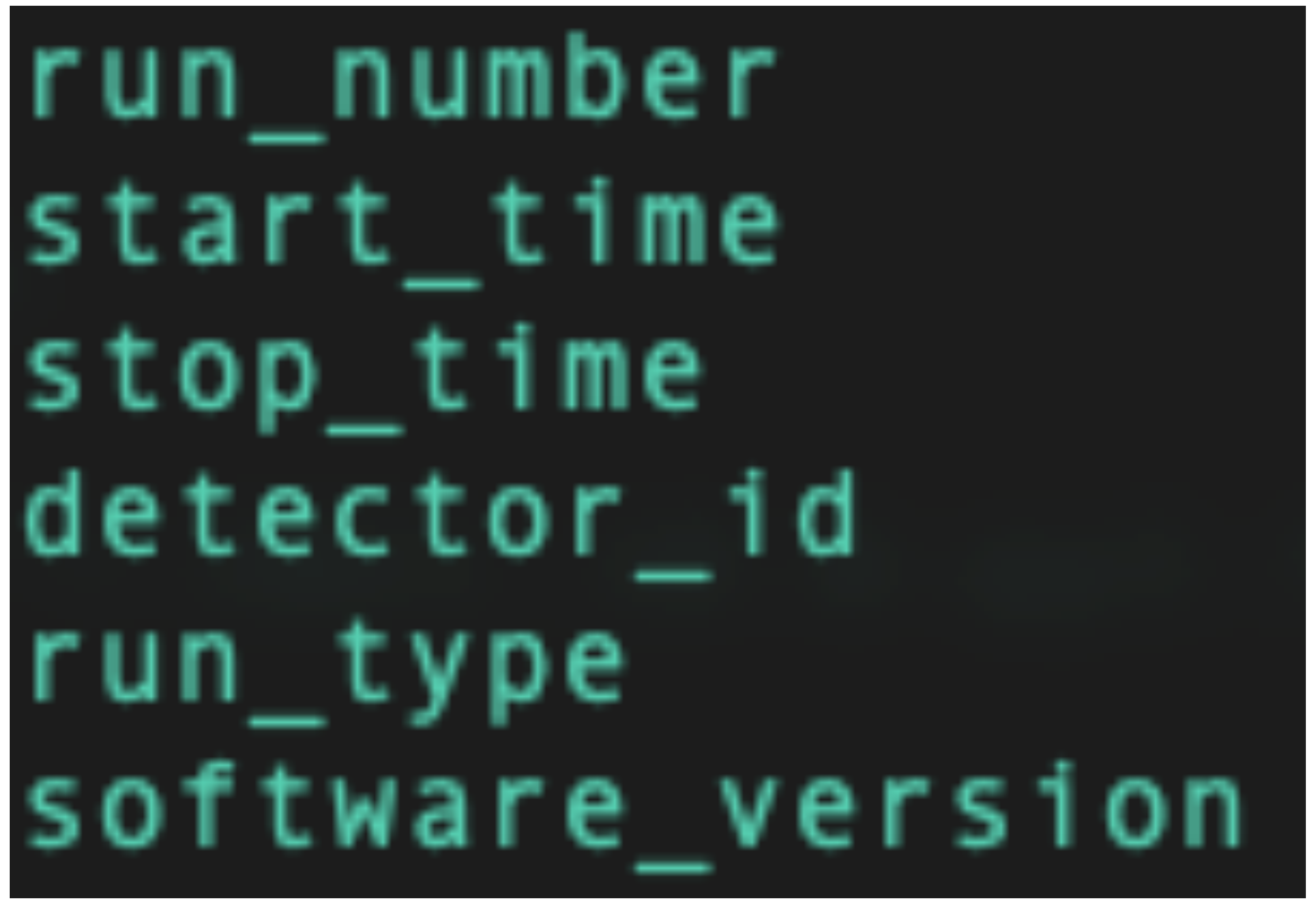
Tables and data

- For a test, the data was loaded to the public schema, but it will be modified in the future
- There is one table with data: test_runs



```
[('test_runs',)]
```

- Data that can be uploaded:



```
run_number
start_time
stop_time
detector_id
run_type
software_version
```

How to access the Run History?

For now

- The data is accessible via Query Engine, which is widely used in protoDUNE

```
-bash-4.2$ curl https://dbdata0vm.fnal.gov:9443/QE/protodune_prod/app/query?t=test_runs
run_number,start_time,stop_time,detector_id,run_type,software_version
12006,2021-11-05 11:31:22-05:00,2021-11-05 11:34:32-05:00,np02_coldbox,PROD,dunedaq-v2.8.1
126,2021-11-05 17:31:22-05:00,2021-11-05 17:34:32-05:00,np02_coldbox,PROD,dunedaq-v2.8.1
1206,2021-11-05 11:31:22-05:00,2021-11-05 11:34:32-05:00,np02_coldbox,PROD,dunedaq-v2.8.1
106,2021-11-05 11:31:22-05:00,2021-11-05 11:34:32-05:00,np02_coldbox,PROD,dunedaq-v2.8.1
```

In the future

- SAM and/or Metacat
- Web interface to see the table with the run history parameters?

New Run History DB for protoDUNE-HD

Location

- It will be a table in the database: pdunehd_prod
- Special permission is needed to access the Db (just for DB experts! - no users)
    - Request access to the DB, create a ticket, and Olga will probably handle it
- Host = ifdbprod.fnal.gov, Port: 5451, and dbname=pdunehd_prod

Amenities

- Development database: pdunehd_dev
    - Used for testing
    - Host = ifdb07.fnal.gov, Port: 5448