Runs database

- Interface

Ana Paula Vizcaya Hernández

Igor Mandrichenko

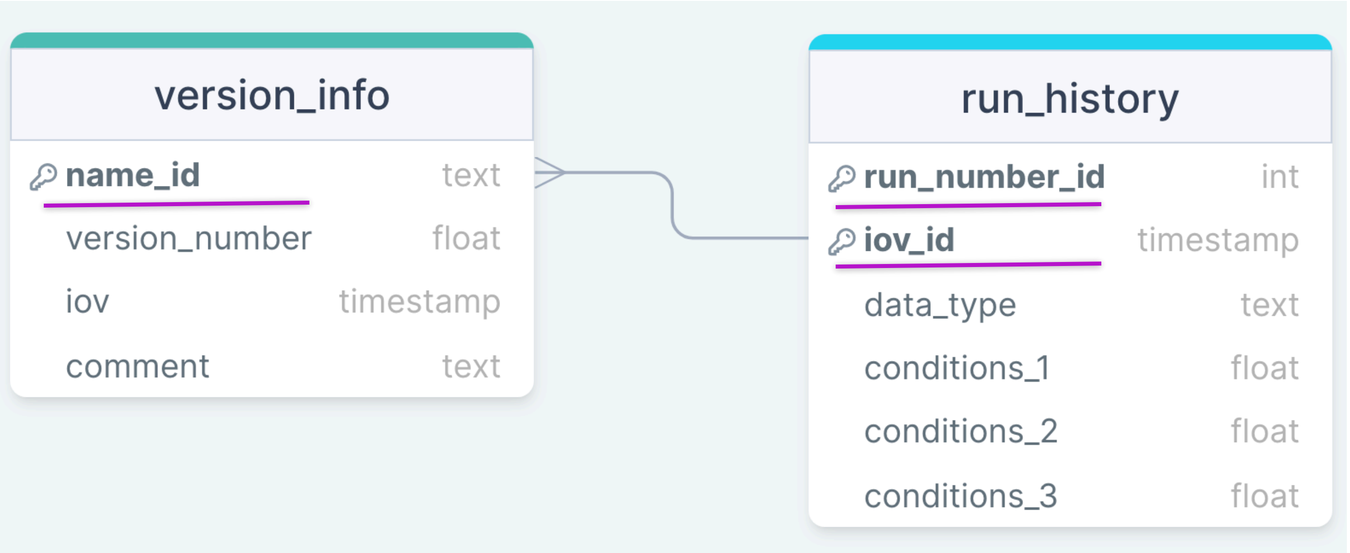
31/10/2023

Colorado State University

Runs Database

https://condb2.readthedocs.io/en/latest/

- Uses the FNAL conditions database interface
- The chosen schema has run number and iov as key parameters. It can also have data_type
- Handles versioning
- More info on (https://condb2.readthedocs.io/en/latest/):
  - How to install
  - Insert data
  - Look for data



Database schema, where the conditions_# payloads represent all the conditions parameters in the database

Create the tables

- Not all users can create tables
  - The area where you can create tables can vary
- What you need to specify:
  - table name = schema.table_name
  - host
  - port
  - user
  - password
  - r and w permissions
  - database
  - columns with the format = column:type

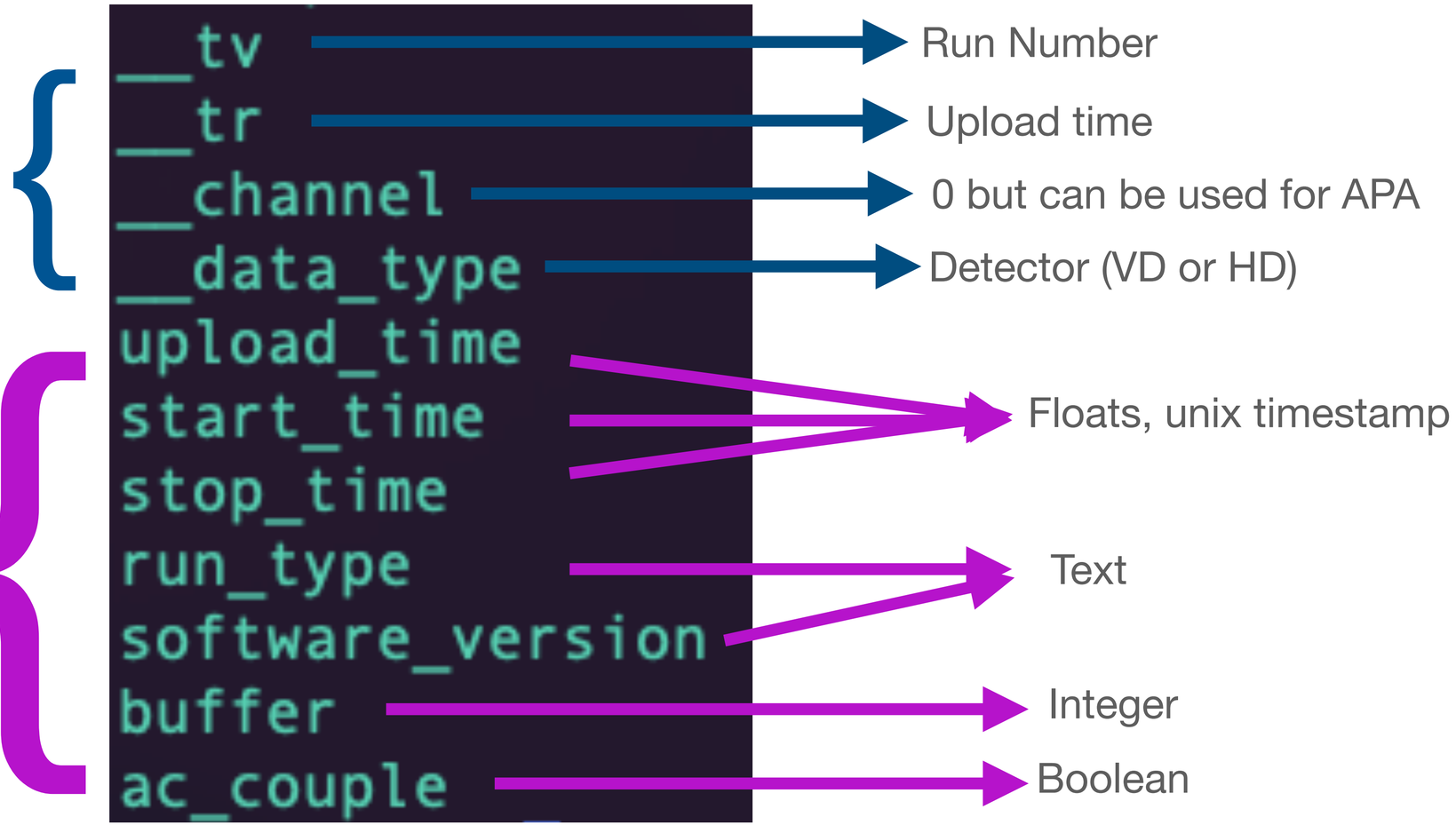
Command line:

```
'condb create -h {host} -p {port} -U {user} -w {passw} -s -R
{r_permission} -W {w_permission} {database} {table_name} {payloads}'
```

Test table

Columns automatically created for all tables

Custom columns



Insert data

https://condb2.readthedocs.io/en/latest/

class condb.**CDFolder**(*db*, *name*, *data_columns_types=None*)

addData(*data*, *data_type=''*, *tr=None*, *columns=None*)

Adds data to the folder

Parameters:
- **data** (*iterable*) – Iterable with tuples: (channel, tv, <data values>, …) channel is the integer channel number tv is numeric validity time (integer or floating point) data values are in the same order as the list of columns used when the folder was created
- **data_type** (*str*) – Data type to associate with the data. Default – blank “”
- **tr** (*float or int*) – Tr to associate the data with. Bt default, current timestamp will be used as floating point number
- **columns** (*list of strings*) – Optional, names of data columns present in the input data, without channel amd tv. If not specified, the data is assumed to contain all the data columns

Returns: Tr timestamp

Return type: float

```
folder.addData(chunk, data_type = self.data.daqmeta_dict['DETECTOR_ID'])
```

Create a version & tag

https://condb2.readthedocs.io/en/latest/

tag(*tag*, *comment*='', *override*=*False*, *tr*=*None*)

Creates new tag with the specified Tr

Parameters:
- **tag** (*str*) – New tag name
- **tr** (*float or int*) – Tr for the tag
- **comment** (*str*) – Comment to add to the new tag
- **override** (*boolean*) – Whether to override an existing tag

- If there are two+ versions of a run the function returns the newest value
- It is possible to extract list of tags

```
folder.tag('v1.2', comment = 'update the buffer 2')
```

Use the searchData() new function

https://condb2.readthedocs.io/en/latest/

searchData(*tag=None, tr=None, data_type=None, channel_range=None, conditions=[]*)

Find all data records on the timeline determined by (tag, tr, data_type)

and satisfying specified conditions expressed in terms of data column values

Parameters:

- **conditions** (*list*) –
  Conditions cpecified as tuples:

  ("column_name", op, value)

  column_name is a name of a data column op is a string "<", "<=", "=", "!=", ">=", ">" value is a string, boolean, numeric or None
- **tr** (*float, int*) – Retieve data retrospectively from a previous state of the database recorded at tr or earlier. By default, will include most recent data.
- **tag** (*str*) – Text tag previously assigned to a Tr value.
- **data_type** (*str*) – Data type to include. If None, will include data for all data types
- **channel_range** (*tuple*) – Tuple (min_channel, max_channel) if provided, only the channels within the specified interval, inclusively will be included in the output. Each one of the limits can be None, which means there is no limit.

Returns: Generator of tuples: (channel, tv, tr, data_type, <data column values>…)

Return type: generator

Use the searchData() SQL part

- The conditional arguments are passed directly to the relational table in sql
- No need to extract all data from db and then do the search
- The rest of the functionalities should also be available

```
            select * from
            (
                select distinct on (u.__channel, u.__tv) u.__channel,u.__tv,u.__tr,u.__data_type,
u.upload_time,u.start_time,u.stop_time,u.run_type,u.software_version,u.buffer,u.ac_couple
                    from %t_update u
                    where
                        (%(tr)s is null or u.__tr < %(tr)s)
                        and (%(data_type)s is null or u.__data_type = %(data_type)s)
                        and (%(min_channel)s is null or u.__channel >= %(min_channel)s)
                        and (%(max_channel)s is null or u.__channel <= %(max_channel)s)
                    order by u.__channel, u.__tv, u.__tr desc
            ) as timeline
            where timeline.run_type = 'PROD' and timeline.stop_time <= '1696849143.0'
        , {'tr': None, 'data_type': 'np04_hd', 'min_channel': None, 'max_channel': None})
(0, 22737.0, 1698716333.2376409, 'np04_hd', 1698716333.2376223, 1695315435.0, 1695315455.0, 'PROD
', 'fddaq-v4.1.1', 2, False)
```

Use the searchData()

One condition

```
con = [('run_type', '=', 'PROD')]
data = self.folder.searchData(tag = '...1.1', conditions=con)
for row in data:
    print(f'{row}')
return data
```

Run number

Condition

```
(0, 22736.0, 1698716333.2174063, 'np02_coldbox', 1698716333.217386, 1695310336.0, 1695...10370.0, 'PROD', ...'fddaq-v4.1.1', 2, False)
(0, 22737.0, 1698716333.2376409, 'np04_hd', 1698716333.2376223, 1695315435.0, 16953154...5.0, 'PROD', 'fddaq-v4.1.1', 2, False)
(0, 22738.0, 1698716333.2572632, 'np02_coldbox', 1698716333.2572448, 1695377434.0, Non..., 'PROD', 'fddaq-v4.1.1', 2, False)
(0, 22746.0, 1698716333.425845, 'np02_coldbox', 1698716333.4258096, 1695585667.0, None..., 'PROD', 'fddaq-v4.1.1', 2, False)
(0, 22747.0, 1698716333.448418, 'np02_coldbox', 1698716333.4484, 1695634894.0, None, '...ROD', 'fddaq-v4...1.1', 2, False)
(0, 22748.0, 1698716333.470041, 'np02_coldbox', 1698716333.4700232, 1695635265.0, None..., 'PROD', 'fddaq-v4.1.1', 2, False)
(0, 22757.0, 1698716333.679413, 'np02_coldbox', 1698716333.6793935, 1696785256.0, None..., 'PROD', 'fddaq-v4.1.1', 2, False)
(0, 22759.0, 1698716333.7307012, 'np02_coldbox', 1698716333.7306466, 1696843781.0, Non..., 'PROD', 'fddaq-v4.1.1', 2, False)
(0, 22760.0, 1698716333.756328, 'np02_coldbox', 1698716333.7563107, 1696844214.0, None..., 'PROD', 'fddaq-v4.1.1', 2, False)
(0, 22761.0, 1698716333.7822492, 'np02_coldbox', 1698716333.7822018, 1696845818.0, Non..., 'PROD', 'fddaq-v4.1.1', 2, False)
(0, 22762.0, 1698716333.8087337, 'np02_coldbox', 1698716333.8086865, 1696846123.0, 169...846144.0, 'PROD', 'fddaq-v4.1.1', 2, False)
(0, 22763.0, 1698716333.8346891, 'np02_coldbox', 1698716333.834671, 1696847379.0, None..., 'PROD', 'fddaq-v4.1.1', 2, False)
(0, 22764.0, 1698716333.8653607, 'np04_hd', 1698716333.8653438, 1696849143.0, 16968491...3.0, 'PROD', 'fddaq-v4.1.1', 2, False)
(0, 22766.0, 1698716333.9114118, 'np04_hd', 1698716333.9113932, 1696851509.0, 16968515...0.0, 'PROD', 'fddaq-v4.1.1', 2, False)
```

Use the searchData()

One condition

```
con = [('run_type', '=', 'TEST')]
data = self.folder.searchData(tag = 'v1.1', conditions=con)
for row in data:
    print(f'{row}')
return data
```

Condition

Run number

```
(0, 22739.0, 1698716333.275736, 'np02_coldbox', 1698716333.2757177, 1695379483.0, 1695379502.0, 'TEST', 'fddaq-v4.1.1', 2, False)
(0, 22740.0, 1698716333.296369, 'np02_coldbox', 1698716333.2963512, 1695379833.0, 1695379856.0, 'TEST', 'fddaq-v4.1.1', 2, False)
(0, 22741.0, 1698716333.320857, 'np02_coldbox', 1698716333.320839, 1695380954.0, 1695380968.0, 'TEST', 'fddaq-v4.1.1', 2, False)
(0, 22742.0, 1698716333.3424945, 'np02_coldbox', 1698716333.3424733, 1695382417.0, 1695382444.0, 'TEST', 'fddaq-v4.1.1', 2, False)
(0, 22743.0, 1698716333.3617685, 'np02_coldbox', 1698716333.3617504, 1695383173.0, 1695383201.0, 'TEST', 'fddaq-v4.1.1', 2, False)
(0, 22744.0, 1698716333.3804367, 'np02_coldbox', 1698716333.3804185, 1695384341.0, 1695384360.0, 'TEST', 'fddaq-v4.1.1', 2, False)
(0, 22745.0, 1698716333.4005396, 'np02_coldbox', 1698716333.4005196, 1695391389.0, 1695391437.0, 'TEST', 'fddaq-v4.1.1', 2, False)
```

Use the searchData()

data_type condition goes in the function as a direct argument

```
con = [('run_type', '=', 'PROD')]
data = self.folder.searchData(conditions=con, data_type="np04_hd")
for row in data:
    print(f'{row}')
return data
```

Run number

```
(0, 22737, 0, 1698716333.2376409, 'np04_hd', 1698716333.2376223, 1695315435.0, 1695315455.0, 'PROD', 'fddaq-v4.1.1', 2, False)
(0, 22764, 0, 1698716333.8653607, 'np04_hd', 1698716333.8653438, 1696849143.0, 1696849163.0, 'PROD', 'fddaq-v4.1.1', 2, False)
(0, 22766, 0, 1698716333.9114118, 'np04_hd', 1698716333.9113932, 1696851509.0, 1696851560.0, 'PROD', 'fddaq-v4.1.1', 2, False)
```

Use the searchData()

Multiple conditions

```
con = [('run_type', '=', 'PROD'), ('stop_time', '<=', 1696849143.0)]
data = self.folder.searchData(conditions=con, data_type="np04_hd")
for row in data:
    print(f'{row}')
return data
```

Run number

Multiple conditions

```
(0, 32737.0, 1698716333.2376409, 'np04_hd', 1698716333.2376223, 1695315435.0, 1695315455.0, 'PROD', 'fddaq-v4.1.1', 2, False)
```

Outlook

- Times are in unix time so I have to add a function to convert to easier (for users) time format
- While I was doing this presentation, the tag function on the searchData function did not work, so look into that
- Add this function to the command line interface?
- Make a git hub folder with example script of these functions
- Make a Jupyter notebook for easier explanation

Thank you

Colorado State University

Backup slides