River Analyst User Manual

River Analyst is a database application framework built with the Django web application framework (Python) to leverage fast river ecosystem analyses.

Installation

Linux

- Clone this repository:

  ```
  git clone https://github.com/beatriznegreiros/river-analyst.git
  ```

- Make sure to have pip3 and virtualenv installed by:

  ```
  sudo apt update
  sudo apt install python3-pip
  pip3 install virtualenv
  ```

- Create new virtual environment:

  ```
  python3.9 -m venv /path/to/new/virtual/environment
  ```

- Activate new virtual environment:

  ```
  source /path/to/new/virtual/environment/bin/activate
  ```

- Install dependencies:

  ```
  pip3 install -r requirements.txt
  ```

Windows

- Clone this repository:

  ```
  git clone https://github.com/beatriznegreiros/river-analyst.git
  ```

- Make sure to have Anaconda installed.
- Create conda environment:

  ```
  conda create --name [env_name] python=3.9
  ```

- Activate conda environment:

  ```
  conda activate [env_name]
  ```

- Install dependencies:

  ```
  pip3 install -r requirements.txt
  ```

Usage

Database architecture

RA database structure is composed of several tables (data models) such as IDO (Interstitial Dissolved Oxygen), which is linked to a MeasStation (measurement stations) via a foreign key. The figure below illustrates the database architecture through an Entity-Relationship diagram:

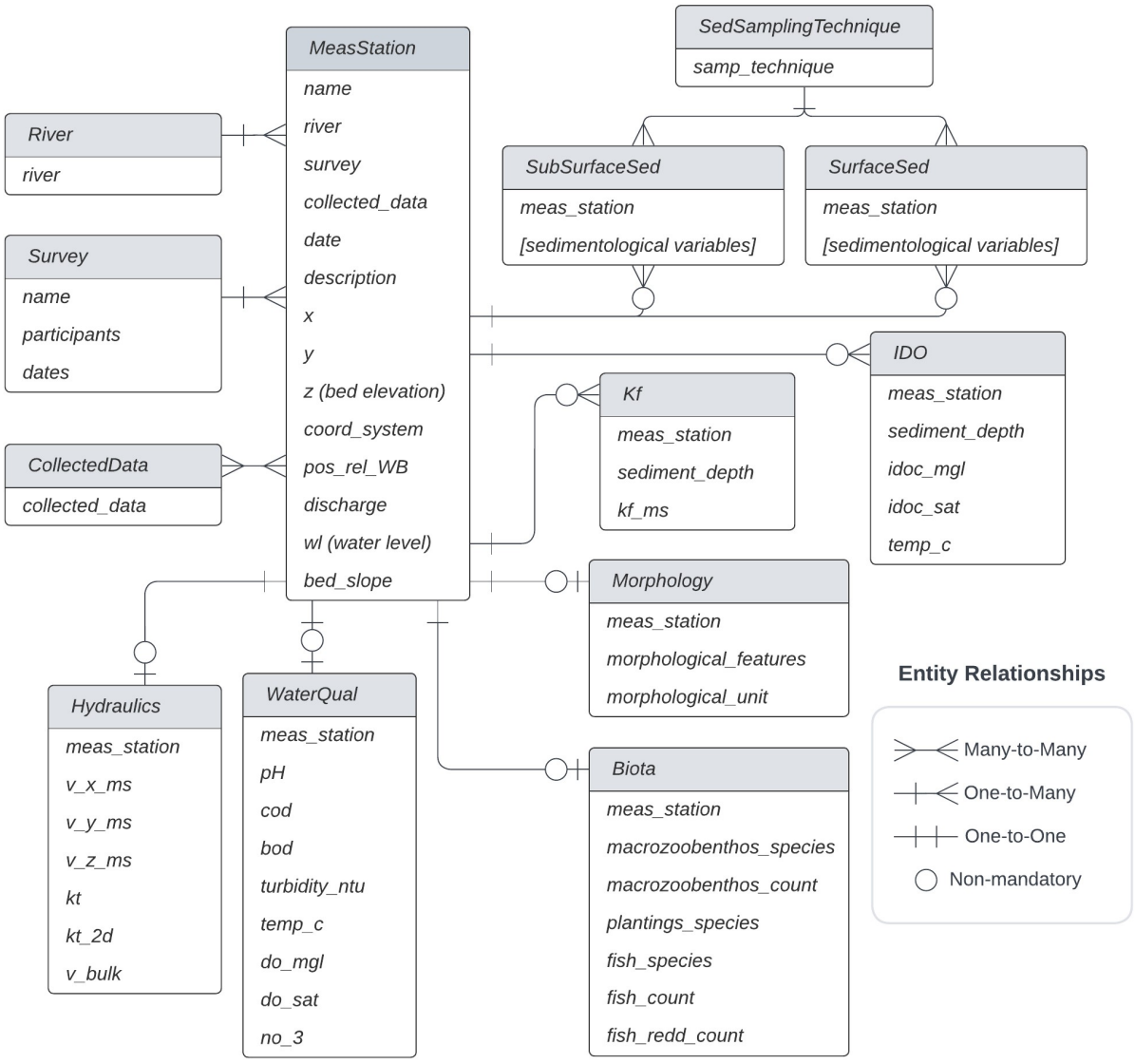

Figure 1: River Analyst database architecture

Figure 2 and 3 provide detailed descriptions of the several attributes within each of RA data models described above.

Running the app

- Go to repository directory

  `cd path/to/river-analyst`

- Make migrations (optional)

  `python3 manage.py migrate` Obs.: Migrations are in principle python commands wrapped around SQL passed from the Django framework to the sql database.

- Run the server locally

  `python3 manage.py runserver`

- Create superuser for having full admin rights over the app:

  `python3 manage.py createsuperuser`

Initializing a new database with template CSVs

- Add data to the csv templates under the path `riveranalyst/river-analyst/media/`
- `cd` to the `riveranalyst/utils` directory `cd riveranalyst/utils`
- Execute scripts to initialize targeted data models
  - It is important to begin with populating the **MeasStation** model, which is where all data models connect:
    - Here, it is crucial that the field `meas_station` is unique and contains no typos. This field will be used to generate foreign keys to link data models. `python fill_stations_tab.py`
  - Then, any data model can be populated afterwards, for instance:
    - the field `meas_station` needs to match the names given in the MeasStation data model.
      - `python fill_surf_tab.py` for filling the **SurfaceSed** data model
      - `python fill_subsurf_tab.py` for the **SubSurfaceSed** data model
      - `python fill_kf_tab.py` for the **Kf** (Riverbed Hydraulic Conductivity) data model
      - `python fill_do_tab.py` for the **IDO** (Interstitial Dissolved Oxygen) data model
      - `python fill_hydraulics_tab.py` for the **Hydraulics** data model

Django cheat sheet (interacting with the Database via Python)

You can create a new Django object by:

```
obj = ModelName(field_name=value)
obj.save()
```

Querying the database is very simple:

```
ModelName.objects.all() # get all objects

# get objects with field_name = value
ModelName.objects.filter(field_name=value)

# get a single object with field_name = value
ModelName.objects.get(field_name=value)
```

Entity		Attributes		
Name	**Description**	**Name**	**Description**	**Instance type**
River		river	River's name	CharField
Survey	Field survey	name	Survey's name	CharField
		participants	Name of field participants	CharField
		start_date	Date on which survey started	DateField
		end_date	Date on which survey ended	DateField
CollectedData	Type of data (e.g., SubsurfaceSed, IDO, Kf, etc)	collected_data	Surveyed river component	CharField with choices
SedSamplTechnique	Sampling technique (for sediment)	samp_techniques	Type of technique (e.g., FC: Freeze Core, OS: Surface Sample)	CharField with choices
MeasStation	A station where one or more measurement procedures were undertaken in an x-y location on a date and time.	name	Station's name	CharField
		river		ForeignKey(River)
		survey		ForeignKey(Survey)
		collected_data		ManyToManyField(CollectedData)
		date	Date of measurement as YYYY-MM-DD	DateField
		description		CharField
		x	X-coordinate (not in degrees)	FloatField
		y	Y-coordinate (not in degrees)	FloatField
		coord_system	epsg projection in which X and Y are	CharField
		x_epsg4326	X-coordinate in EPSG:4326 computed automatically with X and coord_system	FloatField
		y_epsg4326	Y-coordinate in EPSG:4326 computed automatically with Y and coord_system	FloatField
		bed_elevation_wgs84	Ellipsoidal elevation of the riverbed	FloatField
		bed_elevation_dhhn	DHHN (German) elevation of the riverbed	FloatField
		pos_rel_WB	Position relative to the water boundary, "+" for wetted locations	FloatField
		discharge	Flow discharge [m³/s]	FloatField
		wl_m	*in-situ* water depth [m]	FloatField
		wl_model_m	modelled depth level [m]	FloatField
		algae_cover	Presence of algae covering substrate	CharField with choices
		imbrication	Presence of sediment imbrication	CharField with choices
		bed_slope	Bed slope [-]	FloatField
SubsurfaceSed and SurfaceSed	Sedimentological data	meas_station		ForeignKey(MeasStation)
		sample_id	Unique sample name	CharField
		sampling_method		ForeignKey(SedSamplTechnique)
		operator_name	Name of person performing the measurement	CharField
		dm	Mean grain size [mm]	FloatField
		dg	Geometric mean grain size [mm]	FloatField
		fi	Fredle index [mm]	FloatField
		std_grain	Standard deviation of grain sizes [-]	FloatField
		geom_std_grain	Geometric standard deviation of grain sizes [-]	FloatField
		skewness	Skewness of grain size distribution [-]	FloatField
		kurtosis	Kurtosis of grain size distribution [-]	FloatField
		cu	Coefficient of uniformity [-]	FloatField
		cc	Curvature coefficient [-]	FloatField
		n_carling	Porosity according to Carling & Reader (1982)	FloatField
		n_wu_wang	Porosity according to Wu & Wang (2006)	FloatField
		n_wooster	Porosity according to Wooster et al. (2008)	FloatField
		n_frings	Porosity accoridng to Frings et al. (2011)	FloatField
		n_user	Porosity according to Seitz et al. (2018)	FloatField
		d10	Sediment D10 [mm]	FloatField
		d16	Sediment D16 [mm]	FloatField
		d25	Sediment D25 [mm]	FloatField
		d30	Sediment D30 [mm]	FloatField
		d50	Sediment D50 [mm]	FloatField
		d60	Sediment D60 [mm]	FloatField
		d75	Sediment D75 [mm]	FloatField
		d84	Sediment D84 [mm]	FloatField
		d90	Sediment D90 [mm]	FloatField
		so	Sorting coefficient [-]	FloatField
		comment	Comment regarding sample/sampling	CharField
		percent_finer_250mm	Percentage of the sample finer than 250 mm	FloatField
		percent_finer_125mm	Percentage of the sample finer than 125 mm	FloatField
		percent_finer_63mm	Percentage of the sample finer than 63 mm	FloatField
		percent_finer_31_5mm	Percentage of the sample finer than 31.5 mm	FloatField
		percent_finer_16mm	Percentage of the sample finer than 16 mm	FloatField
		percent_finer_8mm	Percentage of the sample finer than 8 mm	FloatField
		percent_finer_4mm	Percentage of the sample finer than 4 mm	FloatField
		percent_finer_2mm	Percentage of the sample finer than 2 mm	FloatField
		percent_finer_1mm	Percentage of the sample finer than 1 mm	FloatField
		percent_finer_0_5mm	Percentage of the sample finer than 0.5 mm	FloatField
		percent_finer_0_25mm	Percentage of the sample finer than 0.25 mm	FloatField
		percent_finer_0_125mm	Percentage of the sample finer than 0.125 mm	FloatField
		percent_finer_0_063mm	Percentage of the sample finer than 0.063 mm	FloatField
		percent_finer_0_031mm	Percentage of the sample finer than 0.031 mm	FloatField

Figure 2: Database attributes Part 1

IDO	Interstitial Dissolved Oxygen	meas_station		ForeignKey(MeasStation)
		sample_id	Unique sample name	CharField
		dp_position	Double packer position ranging from 1 to 15 [-]	IntegerField
		sediment_depth_m	Riverbed/Sediment depth [m]	FloatField
		idoc_mgl	Interstitial dissolved oxygen concentration [mg/L]	FloatField
		idoc_sat	Interstitial dissolved oxygen saturation [%]	FloatField
		temp_c	Interstitial water temperature [°C]	FloatField
		H_m	Heigh of filter pipe above riverbed [m]	FloatField
		operator_name	Name of person performing the measurement	CharField
		comment	Comment regarding the measurement	Charfield
Kf	Hydraulic Conductivity and suction tests data	meas_station		ForeignKey(MeasStation)
		sample_id	Unique sample name	CharField
		dp_position	Double packer position ranging from 1 to 15 [-]	IntegerField
		sediment_depth_m	Riverbed/Sediment depth [m]	FloatField
		kf_ms	Hydraulic Conductivity [m/s]	FloatField
		slurp_rate_avg_mls	Slurping rate [ml/s]	FloatField
		H_m	Heigh of filter pipe above riverbed [m]	FloatField
		operator_name	Name of person performing the measurement	CharField
		comment	Comment regarding the measurement	CharField
Hydraulics	Free-flow hydraulic data	meas_station		ForeignKey(MeasStation)
		sample_id	Unique sample name	CharField
		v_x_ms	Longitudinal velocity component [m/s]	FloatField
		v_y_ms	Lateral velocity component [m/s]	FloatField
		v_z_ms	Vertical velocity component [m/s]	FloatField
		kt	Turbulent kinetic energy in x, y, and z [m^2/s^2]	FloatField
		kt_2d	Turbulent kinetic energy in x, and y [m^2/s^2]	FloatField
		v_bulk	Bulk flow velocity [m/s]	FloatField
		water_temperature	Free-flowing-water temperature [°C]	FloatField
		operator_name	Name of person performing the measurement	CharField
		comment	Comment regarding the measurement	CharField
		ship_influence	Presence of ship influence in the form of water level fluctuations	CharField with multiple choice
WaterQual	Water quality data	meas_station		ForeignKey(MeasStation)
		sample_id	Unique sample name	CharField
		ph	pH [-]	FloatField
		cod	COD [mg/L]	FloatField
		bod	BOD [mg/L]	FloatField
		turbidity_ntu	Turbidity [NTU]	FloatField
		temp_c	Temperature [°C]	FloatField
		do_mgl	Dissolved oxygen concentration [mg/L]	FloatField
		do_sat	Dissolved oxygen saturation [%]	FloatField
		no_3	Nitrate (NO-3) concentration [mg/L]	FloatField
Biota	Biotic attributes	meas_station		ForeignKey(MeasStation)
		sample_id	Unique sample name	CharField
		macrozoobenthos_species	Species of macrozoobenthos found, use comma to list more than one species	CharField
		macrozoobenthos_count	Number of macrozoobenthos found, use comma to list more than one species	CharField
		planting_species	Plantings species observed	CharField
		fish_species	Fish species observed, use comma to list more than one species	CharField
		fish_redd_count	Number of fish redds observed, use comma to list more than one species	CharField
Morphology	Morphological attributes	meas_station		ForeignKey(MeasStation)
		sample_id	Unique sample name	CharField
		morph_features	Morphological features (e.g., Wood logs)	CharField
		morph_unit	Morphological unit (e.g., Riffle)	CharField

Figure 3: Database attributes Part 2

To create a new Django model, you need to define a class in one of your Django app's `models.py` file that inherits from Django's built-in models.Model class. Here is an example model class that defines a Book model with fields for title, author, and publication date:

```
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=200)
    author = models.CharField(max_length=200)
    pub_date = models.DateField()
```

Connecting the project with a database file stored in the cloud (Example for AWS RDS)

- Install the psycopg2 library: Since AWS RDS supports PostgreSQL, you will need to install the psycopg2 library, which is a PostgreSQL adapter for Python, by running the following command:

```
pip install psycopg2-binary
```

- Configure the Django project settings: In your Django project's settings.py file, you will need to configure the database settings to connect to your AWS RDS instance. Here is an example configuration for a PostgreSQL database:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'your-db-name',
        'USER': 'your-db-username',
        'PASSWORD': 'your-db-password',
        'HOST': 'your-db-endpoint.aws-region.rds.amazonaws.com',
        'PORT': '5432',
    }
}
```

In the above configuration, you will need to replace `your-db-name`, `your-db-username`, and `your-db-password` with your own values, and replace `your-db-endpoint` and `aws-region` with the endpoint and region of your AWS RDS instance, respectively. You can find your RDS instance's endpoint in the RDS console.

- Migrate the Django project: Once you have configured your database settings, you will need to run the following commands to migrate the Django project to the database:

```
python manage.py makemigrations
python manage.py migrate
```

These commands will create the necessary tables and columns in your database.

- Test the connection: Finally, you can test the connection to your AWS RDS instance by running the following command:

```
python manage.py dbshell
```

This command will open a PostgreSQL shell that connects to your database. If the connection is successful, you should see a prompt that looks like this:

```
psql (13.4, server 13.3)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.
```

```
your-db-name=>
```