ActInsight Data-Sharing API

Quick Start

ValueGrid

October 13th, 2022

Introduction

Why?

As part of the ActInsight solution, in the context of the Global Stocktake Climate Datathon, we wrote an API to provide access to the data-layer of our wider application.

To fully leverage climate data interoperability, we built our backend database (PostgreSQL 14) stricly following the OpenClimate Schema, designed & maintained by the OpenEarth Foundation team.

If you also follow this OpenClimate standard, all data from the API calls (as defined in following sections) should be a direct drop-in in your existing data flows. Even if you have your own custom schema, you can benefit by pulling raw data to enrich your datasets as needed.

All data is publicly available but pulling it and structuring it from different institutional websites was tedious, hence the work proposed here with a streamlined collaborative easy-to-use API.

How does it work?

Access

The REST API we built is completely language agnostic, meaning that you can access it from any kind of backend/frontend stack.

If you are not too familiar with programming, you can simply export CSV sheets from the ActInsight website. Alternatively, for full impact to your own solutions, best is to pull it directly from your favorite stack. In the following sections, we show you how to bring data directly in your **R** or Python workflows, as data frames, which we believe is fairly sweet and efficient, so that you can focus on the added-value on top of this data layer.

Current scope

Just to give an idea/scale of the current content (as of October 13th, 2022), you can get relational data for ~150 climate initiatives, 10,319 companies, 1,523 investors, 762 climate actions, 44 sectors, 1,187 emissions reduction plans, 3,256 institutional organizations, 266 regions, 196 countries, 10,210 cities (with corresponding territorial information), 1,916 emissions records & recorded targets.

As for the roadmap ahead, we will follow the Community consensus.

Data sources

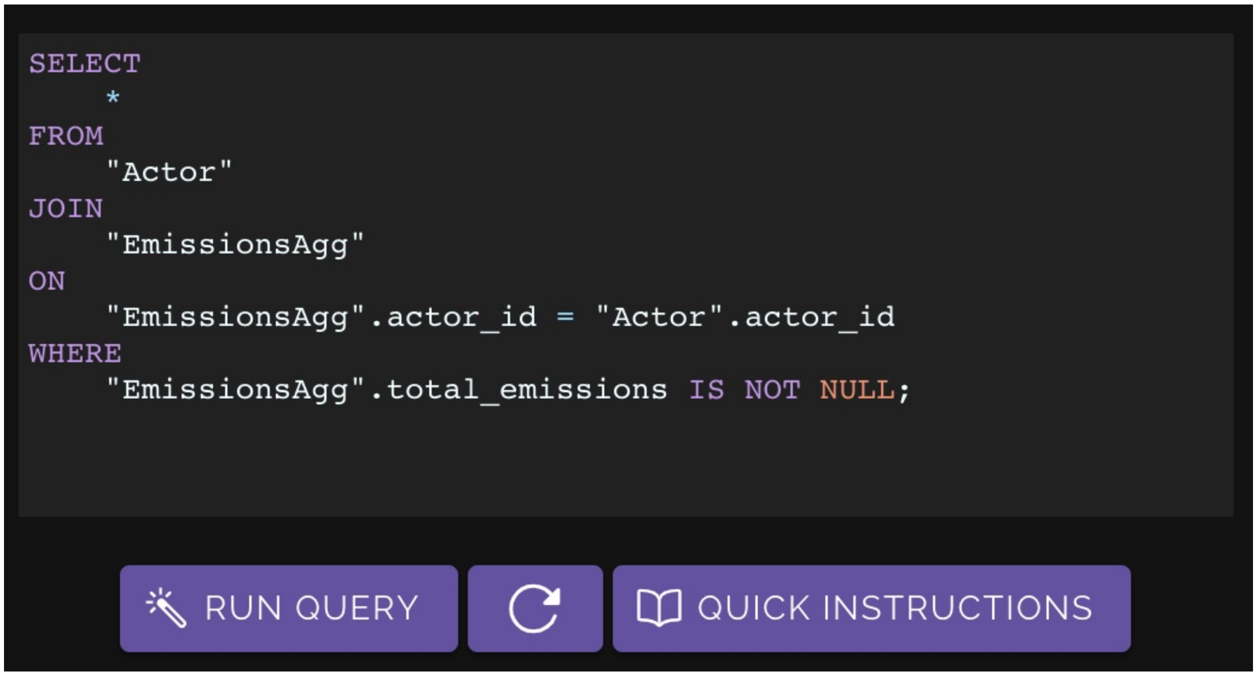
The current input data sources are the following ones:

- UNFCCC Climate Action
- Net Zero Tracker

- Global Covenant of Mayors
- WikiData pySPARK custom queries

Access from the Web

Head to https://ActInsight.org/data



```sql
SELECT
    *
FROM
    "Actor"
JOIN
    "EmissionsAgg"
ON
    "EmissionsAgg".actor_id = "Actor".actor_id
WHERE
    "EmissionsAgg".total_emissions IS NOT NULL;
```

Figure 1: Enter your custom SQL queries directly from the ActInsight web interface and download CSV output results as needed

Access from R

```r
library(httr) # for our HTTPS calls
library(kableExtra) # optional, just for nicer table formatting

# our API base URL
queryApiBaseUrl <- 'https://us-central1-actinsightorg.cloudfunctions.net/data-sharing'

# Your own SQL SELECT statement, just an example here
myQuery <- 'SELECT *
            FROM "Actor"
            JOIN "EmissionsAgg"
            ON "EmissionsAgg".actor_id = "Actor".actor_id
            WHERE "EmissionsAgg".total_emissions IS NOT NULL;'

# build HTTP parameters
params <- list(type="query", query=myQuery)

# make the HTTPS GET call
```

```r
r <- httr::GET(queryApiBaseUrl, query=params)

# check if returned code is OK
httr::status_code(r) # 200 OK
```

```
## [1] 200
```

```r
# get response content and pass it straight to a dataframe that you can then easily work with
df <- httr::content(r)
```

```
## New names:
## Rows: 106 Columns: 18
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (12): actor_id...1, type, name, icon, hq, is_part_of, is_owned_by, data... dbl
## (2): year, total_emissions dttm (2): created...17, last_updated...18 date (2):
## created...9, last_updated...10
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `actor_id` -> `actor_id...1`
## * `datasource_id` -> `datasource_id...8`
## * `created` -> `created...9`
## * `last_updated` -> `last_updated...10`
## * `actor_id` -> `actor_id...12`
## * `datasource_id` -> `datasource_id...16`
## * `created` -> `created...17`
## * `last_updated` -> `last_updated...18`
```

```r
# optional, drop some of the columns for better table visibility
drops <- c("icon", "hq", "is_part_of", "is_owned_by", "datasource_id...8", "created...9", "last_updated
df <- df[ , !(names(df) %in% drops)]

# show start of the dataframe
kableExtra::kbl(head(df, 15), booktabs=T)
```

actor_id...1	type	name	year	total_emissions
OC_ACTOR_36	city	Accra	2020	2321904
OC_ACTOR_62	city	Addis Ababa	2021	9703285
OC_ACTOR_184	city	Albany NY	2012	996818
OC_ACTOR_424	city	Amman	2020	9656048
OC_ACTOR_428	city	Amsterdam	2020	4803879
OC_ACTOR_677	city	Athens	2019	1414610
OC_ACTOR_693	city	Austin TX	2021	11965153
OC_ACTOR_805	city	Baltimore MD	2019	5489334
OC_ACTOR_853	city	Barcelona	2018	2784868
OC_ACTOR_892	city	Basel	NA	825703
OC_ACTOR_989	city	Belo Horizonte	2021	4398073
OC_ACTOR_1085	city	Berlin	2021	9607000
OC_ACTOR_1170	city	Birmingham UK	2021	3070431
OC_ACTOR_1219	city	Bogor	NA	1472574
OC_ACTOR_1220	city	Bogotá	2020	11421723

Access from Python

```
import requests # for our HTTPS calls
import pandas as pd # Pandas for dataframes, what else:)
from io import StringIO # to treat the CSV format directly in memory

# our API base URL
query_api_base_url = 'https://us-central1-actinsightorg.cloudfunctions.net/data-sharing'

# Your own SQL SELECT statement, just an example here
my_query ="""SELECT *
             FROM "Actor"
             JOIN "EmissionsAgg"
             ON "EmissionsAgg".actor_id = "Actor".actor_id
             WHERE "EmissionsAgg".total_emissions IS NOT NULL;"""

# build HTTP parameters
payload = {'type': 'query', 'query': my_query}

# make the HTTPS GET call
r = requests.get(query_api_base_url, params=payload)

# get response content and pass it straight to a dataframe that you can then easily work with
csv_string_io = StringIO(r.text)
df = pd.read_csv(csv_string_io)

# optional, drop some of the columns for better table visibility
df = df.drop(columns=['icon', 'hq', 'is_part_of', 'is_owned_by', 'datasource_id', 'created',
                      'last_updated', 'emissions_id', 'actor_id.1', 'methodology_id',
                      'datasource_id.1', 'created.1', 'last_updated.1'])

# show start of the dataframe
df.head(15)
```

```
##          actor_id  type           name    year  total_emissions
## 0     OC_ACTOR_36  city          Accra  2020.0          2321904
## 1     OC_ACTOR_62  city    Addis Ababa  2021.0          9703285
## 2    OC_ACTOR_184  city      Albany NY  2012.0           996818
## 3    OC_ACTOR_424  city          Amman  2020.0          9656048
## 4    OC_ACTOR_428  city      Amsterdam  2020.0          4803879
## 5    OC_ACTOR_677  city         Athens  2019.0          1414610
## 6    OC_ACTOR_693  city      Austin TX  2021.0         11965153
## 7    OC_ACTOR_805  city    Baltimore MD  2019.0          5489334
## 8    OC_ACTOR_853  city      Barcelona  2018.0          2784868
## 9    OC_ACTOR_892  city          Basel     NaN           825703
## 10   OC_ACTOR_989  city  Belo Horizonte  2021.0          4398073
## 11  OC_ACTOR_1085  city         Berlin  2021.0          9607000
## 12  OC_ACTOR_1170  city   Birmingham UK  2021.0          3070431
## 13  OC_ACTOR_1219  city          Bogor     NaN          1472574
## 14  OC_ACTOR_1220  city         Bogotá  2020.0         11421723
```

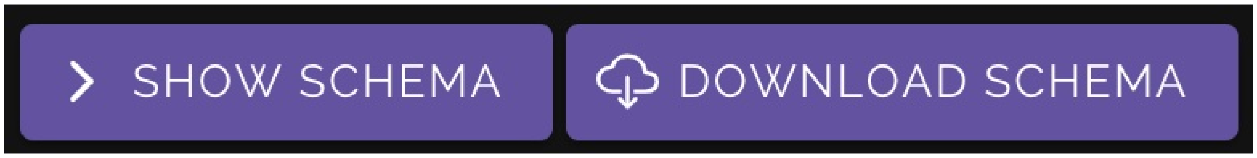
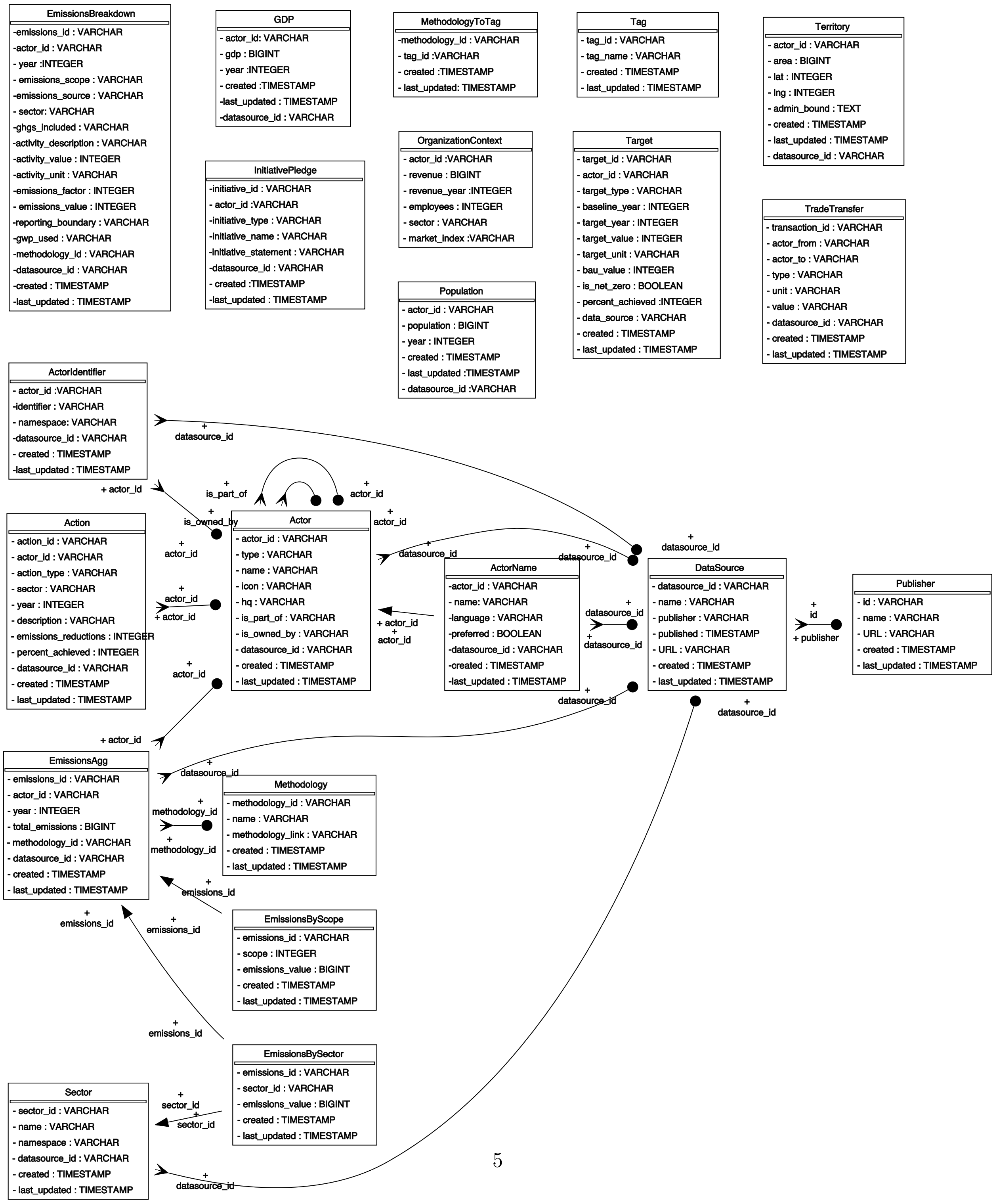


Figure 2: Latest generated data model available on the ActInsight website

Current database data model


EmissionsBreakdown
-emissions_id : VARCHAR
-actor_id : VARCHAR
- year :INTEGER
- emissions_scope : VARCHAR
-emissions_source : VARCHAR
- sector: VARCHAR
-ghgs_included : VARCHAR
-activity_description : VARCHAR
-activity_value : INTEGER
-activity_unit : VARCHAR
-emissions_factor : INTEGER
- emissions_value : INTEGER
-reporting_boundary : VARCHAR
-gwp_used : VARCHAR
-methodology_id : VARCHAR
-datasource_id : VARCHAR
-created : TIMESTAMP
-last_updated : TIMESTAMP
GDP
- actor_id: VARCHAR
- gdp : BIGINT
- year :INTEGER
- created :TIMESTAMP
-last_updated : TIMESTAMP
-datasource_id : VARCHAR
MethodologyToTag
-methodology_id : VARCHAR
- tag_id :VARCHAR
- created :TIMESTAMP
- last_updated: TIMESTAMP
Tag
- tag_id : VARCHAR
- tag_name : VARCHAR
- created : TIMESTAMP
- last_updated : TIMESTAMP
Territory
- actor_id : VARCHAR
- area : BIGINT
- lat : INTEGER
- lng : INTEGER
- admin_bound : TEXT
- created : TIMESTAMP
- last_updated : TIMESTAMP
- datasource_id : VARCHAR
InitiativePledge
-initiative_id : VARCHAR
- actor_id :VARCHAR
-initiative_type : VARCHAR
-initiative_name : VARCHAR
-initiative_statement : VARCHAR
-datasource_id : VARCHAR
- created :TIMESTAMP
-last_updated : TIMESTAMP
OrganizationContext
- actor_id :VARCHAR
- revenue : BIGINT
- revenue_year :INTEGER
- employees : INTEGER
- sector : VARCHAR
- market_index :VARCHAR
Target
- target_id : VARCHAR
- actor_id : VARCHAR
- target_type : VARCHAR
- baseline_year : INTEGER
- target_year : INTEGER
- target_value : INTEGER
- target_unit : VARCHAR
- bau_value : INTEGER
- is_net_zero : BOOLEAN
- percent_achieved :INTEGER
- data_source : VARCHAR
- created : TIMESTAMP
- last_updated : TIMESTAMP
TradeTransfer
- transaction_id : VARCHAR
- actor_from : VARCHAR
- actor_to : VARCHAR
- type : VARCHAR
- unit : VARCHAR
- value : VARCHAR
- datasource_id : VARCHAR
- created : TIMESTAMP
- last_updated : TIMESTAMP
Population
- actor_id : VARCHAR
- population : BIGINT
- year : INTEGER
- created : TIMESTAMP
- last_updated :TIMESTAMP
- datasource_id :VARCHAR
ActorIdentifier
- actor_id :VARCHAR
-identifier : VARCHAR
- namespace: VARCHAR
-datasource_id : VARCHAR
- created : TIMESTAMP
-last_updated : TIMESTAMP
+ datasource_id
+ actor_id
+ is_part_of
+ actor_id
+ is_owned_by
+ actor_id
Action
- action_id : VARCHAR
- actor_id : VARCHAR
- action_type : VARCHAR
- sector : VARCHAR
- year : INTEGER
- description : VARCHAR
- emissions_reductions : INTEGER
- percent_achieved : INTEGER
- datasource_id : VARCHAR
- created : TIMESTAMP
- last_updated : TIMESTAMP
Actor
- actor_id : VARCHAR
- type : VARCHAR
- name : VARCHAR
- icon : VARCHAR
- hq : VARCHAR
- is_part_of : VARCHAR
- is_owned_by : VARCHAR
- datasource_id : VARCHAR
- created : TIMESTAMP
- last_updated : TIMESTAMP
+ actor_id
+ actor_id
+ actor_id
+ actor_id
+ datasource_id
+ datasource_id
+ actor_id
+ actor_id
ActorName
-actor_id : VARCHAR
- name: VARCHAR
-language : VARCHAR
-preferred : BOOLEAN
-datasource_id : VARCHAR
-created : TIMESTAMP
-last_updated : TIMESTAMP
+ datasource_id
+ datasource_id
+ datasource_id
+ datasource_id
DataSource
- datasource_id : VARCHAR
- name : VARCHAR
- publisher : VARCHAR
- published : TIMESTAMP
- URL : VARCHAR
- created : TIMESTAMP
- last_updated : TIMESTAMP
+ id
+ publisher
Publisher
- id : VARCHAR
- name : VARCHAR
- URL : VARCHAR
- created : TIMESTAMP
- last_updated : TIMESTAMP
+ datasource_id
+ datasource_id
+ actor_id
EmissionsAgg
- emissions_id : VARCHAR
- actor_id : VARCHAR
- year : INTEGER
- total_emissions : BIGINT
- methodology_id : VARCHAR
- datasource_id : VARCHAR
- created : TIMESTAMP
- last_updated : TIMESTAMP
+ datasource_id
+ methodology_id
+ methodology_id
Methodology
- methodology_id : VARCHAR
- name : VARCHAR
- methodology_link : VARCHAR
- created : TIMESTAMP
- last_updated : TIMESTAMP
+ emissions_id
+ emissions_id
+ emissions_id
EmissionsByScope
- emissions_id : VARCHAR
- scope : INTEGER
- emissions_value : BIGINT
- created : TIMESTAMP
- last_updated : TIMESTAMP
+ emissions_id
EmissionsBySector
- emissions_id : VARCHAR
- sector_id : VARCHAR
- emissions_value : BIGINT
- created : TIMESTAMP
- last_updated : TIMESTAMP
Sector
- sector_id : VARCHAR
- name : VARCHAR
- namespace : VARCHAR
- datasource_id : VARCHAR
- created : TIMESTAMP
- last_updated : TIMESTAMP
+ sector_id
+ sector_id
+ datasource_id