EM-DAT GraphQL API Cookbook

Current version metadata

Retrieving the latest metadata of the can be done this way. Note that the `total_available` field will only count non-historical entries (from 2000 onwards).

Queries can also be commented using the `#` character. With this character, the rest of the line is simply ignored. This is helpful for documenting, explaining what is extracted for later or for ignoring a field in the request execution.

Keeping track of the query metadata can help for storing and comparing the data retrieved and avoid over-fetching : if the field `public_emdat.info.version` is unchanged, the data available has not changed. The `filters` also helps in keeping track of what was exactly queried or to compare how the `filters` argument on `public_emdat` was eventually processed.

```
query metadata_only {
    api_version             # the version of the API accessed
    public_emdat {
        total_available     # total count of events
        info {
            timestamp       # when was the query executed
            version         # latest update of in the whole public table
            # The 2 fields below are ignored as they are empty JSON Objects.
            # They reflect what was passed to *public_emdat* arguments.
            # filters
            # cursor
        }
    }
}
```

Belgian floods example

We will now posit a specific example and modify it in different places, affecting the result. Our base case is to lookup floods occurring in Belgium and more specifically look at:

1. Disaster identification with the disaster number (DisNo. and Classification key).
2. their temporality, start and end date when possible
3. the river basin or location as names if they are mentioned
4. the administrative regions affected
5. the impact values we want to collect in terms of economic damage and number of people affected
6. entry tracking for creation and latest update

```
query floods_bel {
    api_version
    public_emdat(filters: {
        iso: ["BEL"],               # Belgium only
        classif: ["nat-hyd-flo-*"]  # Classification wildcard
        # ==> taking all entries of the Type "Flood"
    }) {
        total_available
        info {
            timestamp
            version
            filters
            cursor
        }
        data {
            disno               # 1. identification
            classif_key
            start_year          # 2. temporality
            start_month
            start_day
            end_year
            end_month
            end_day
            river_basin         # 3. text location description
            location
            admin_units         # 4. structured locations
            total_dam           # 5. impact values
            total_dam_adj
            total_affected
            entry_date          # 6. tracking
            last_update
        }
    }
}
```

Note first that the result fields will be present in the result JSON in the order specified.

Building up: historical values, adding countries, taking all

The `public_emdat` query has 2 implicit limitations activated by default. These are meant to ease the interactive development of queries in GraphiQL and can be overridden easily when moving to the actual data extraction.

1. Historical entries (before 2000) are filtered out

For reasons of methodological change, old entries are ignored by default and left out of the result. It is possible to include them with the following modification:

```
query floods_bel {
    api_version
    public_emdat(filters: {
```

```
        iso: ["BEL"]
        classif: ["nat-hyd-flo-*"]
        include_hist: true
    }) {
        # ...metadata
        data {
            disno
            classif_key
            # ...other fields
        }
    }
}
```

2. Implicit limit of request size

The number of entries is limited by default, one can either scroll through the following entries by incrementally store entries, using the `offset` field of the `cursor` argument. This could be helpful to implement a table pagination but the limitation can be overridden altogether with the `limit` field on the same argument. One can either put an upper bound limit to a specific number or remove it by setting it to the value `-1`. This last option is important when moving to analysis after the musing/mocking stage but can make GraphiQL pretty slow with limited benefit.

```
query floods_bel {
    api_version
    public_emdat(filters: {
        iso: ["BEL"]
        classif: ["nat-hyd-flo-*"]
        include_hist: true
    }, cursor: {
        offset: 10 # ignore the first 10 entries
        limit: -1   # unlimited query
    }) {
        # ...metadata
        data {
            disno
            classif_key
            # ...other fields
        }
    }
}
```