Construcción de URLs para queries en Tesseract OLAP

Datawheel, LLC

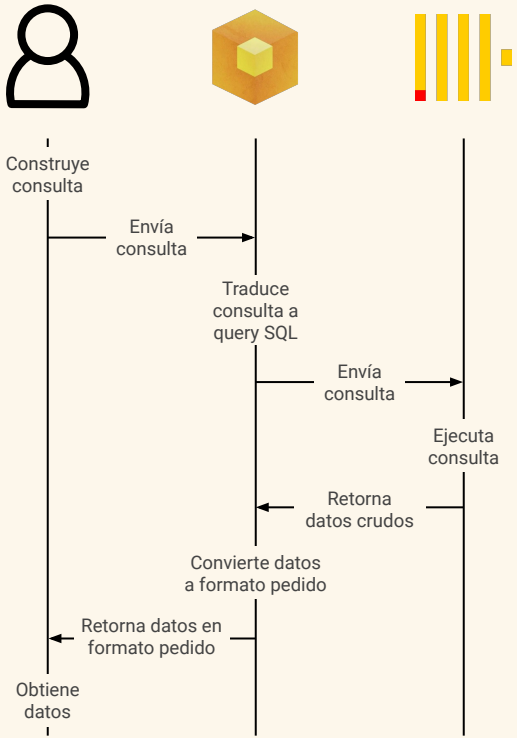
Septiembre 2020

Tesseract OLAP

Es un servidor ROLAP (**R**elational **O**n**l**ine **A**nalytical **P**rocessing) de código abierto escrito en Rust.
El código fuente es mantenido en https://github.com/tesseract-olap/tesseract

A partir de un esquema de datos, traduce consultas por información en queries SQL, que son ejecutadas por Clickhouse.

El resultado de la consulta es convertido a formato JSONRecords (Tidy data en JSON), JSONArray (CSV preparseado) o CSV crudo, y devuelto al usuario.



Esquema de datos

Tesseract sigue un esquema para validar la consulta antes de enviarla a Clickhouse. Es por eso que la petición debe construirse utilizando datos del esquema que tesseract hace público.

El esquema se define internamente en un archivo XML con la información de las tablas y columnas de datos en Clickhouse, y cómo están asociadas a dimensiones y métricas en la abstracción de cubos. El usuario final sólo debe preocuparse de lo último al momento de construir la consulta.

```
$ curl -i https://api.datamexico.org/tesseract/cubes

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 296245
Access-Control-Allow-Origin: *

{
  "name": "datamexico",
  "cubes": [
    {
      "name": "anuies_enrollment",
      "measures": [
        {
          "name": "Students",
          "aggregator": {"name": "sum"},
          "measure_type": {
            "standard": {"units": null}
          },
          "annotations": {...}
        }
      ],
      "dimensions": [
        {
          "name": "Geography",
          "default_hierarchy": "Geography",
          "type": "geo",
          "annotations": {},
          "hierarchies": [
            {
              "name": "Geography",
              "levels": [
                {
                  "name": "Nation",
                  "properties": [...],
                  "annotations": {},
                  "unique_name": null
                },
                {
                  "name": "State",
                  "properties": [...],
                  "annotations": {},
                  "unique_name": null
                },
                ...
              ],
              "annotations": {...}
            },
...
```

```
$ SCHEMA=`curl https://api.datamexico.org/
tesseract/cubes`
$ echo $SCHEMA | jq '.'

{
  "name": "datamexico",
  "cubes": [...],
  "annotations": {
    "available_languages": "es",
  }
}
```

Estructura

El esquema general que devuelve Tesseract contiene un esquema bien definido, que aplica para todos sus elementos.

El elemento raíz puede representarse de esta forma:

```
interface Schema {
  name: string;
  cubes: Cube[];
  annotations: Annotations;
}

type Annotations = {
  [key: string]: string;
};
```

```
$ echo $SCHEMA | jq '.cubes[0]'

{
  "name": "anuies_enrollment",
  "annotations": {
    "topic": "Education",
    "subtopic": "Enrollment",
    "source_name": "Asociación Nacional …",
    "source_name_en": "National Associat…",
    "source_link": "http://www.anuies.mx…"
  },
  "dimensions": [
    {
      "name": "Geography",
      "hierarchies": [...],
      "default_hierarchy": "Geography",
      "type": "geo",
      "annotations": {}
    },
    ...
  ],
  "measures": [
    {
      "name": "Students",
      "aggregator": {
        "name": "sum"
      },
      "measure_type": {
        "standard": {"units": null}
      },
      "annotations": {
        "details": "Number of Students",
      }
    }
  ]
}
```

Estructura

Un cubo contiene información relacionada a un mismo tema. El cubo anuies_enrollment sólo contiene información sobre matrícula, provista por la ANUIES.

```
interface Cube {
  name: string;
  annotations: Annotations;
  dimensions: Dimension[];
  measures: Measure[];
}
```

Los datos almacenados en el cubo pueden ser agrupados según sus *dimensiones*, y agregar sus *métricas* para obtener la información que necesitamos.

```
$ echo $SCHEMA | jq '.cubes[0]'

{
  "name": "anuies_enrollment",
  "annotations": {...},
  "dimensions": [...],
  "measures": [
    {
      "name": "Students",
      "aggregator": {
        "name": "sum"
      },
      "measure_type": {
        "standard": {
          "units": null
        }
      },
      "annotations": {
        "details": "Number of Students",
      }
    }
  ]
}
```

Estructura

Las métricas (measures) de un cubo son los valores observados para cada medición realizada.

```
interface Measure {
  annotations: Annotations;
  name: string;
  aggregator: {
    name: "avg" | "count" | "max" | "min" | "sum" | [...];
  }
  measure_type: MeasureStandard | MeasureError;
}
```

```
type MeasureStandard = {
  standard: {
    units: string;
  }
}
```

```
type MeasureError = {
  error: {
    for_measure: string;
    err_type: string;
  }
}
```

```
$ echo $SCHEMA | jq '.cubes[0]'

{
  "name": "anuies_enrollment",
  "annotations": {...},
  "dimensions": [
    {
      "name": "Geography",
      "hierarchies": [...],
      "default_hierarchy": "Geography",
      "type": "geo",
      "annotations": {}
    },
    {
      "name": "Sex",
      "hierarchies": [...],
      "default_hierarchy": null,
      "type": "standard",
      "annotations": {}
    },
    {
      "name": "Year",
      "hierarchies": [...],
      "default_hierarchy": null,
      "type": "time",
      "annotations": {}
    },
    [...]
  ],
  "measures": [...]
}
```

Estructura

Las dimensiones (dimensions) son las variables asociadas a la situación en que las métricas fueron obtenidas.

```
interface Dimension {
  annotations: Annotations;
  name: string;
  hierarchies: Hierarchy[];
  default_hierarchy?: Hierarchy[];
  type: "geo" | "standard" | "time";
}
```

Una dimensión agrupa variables de un mismo tipo. Estas variables pueden ser categorizadas en jerarquías, cada una con sus propios niveles de detalle.

```
$ echo $SCHEMA | \
jq '.cubes[0].dimensions[0]'

{
  "name": "Geography",
  "hierarchies": [
    {
      "name": "Geography",
      "levels": [...],
      "annotations": {}
    },
    {
      "name": "Metro Area",
      "levels": [...],
      "annotations": {}
    },
    {
      "name": "National Urban System",
      "levels": [...],
      "annotations": {}
    }
  ],
  "default_hierarchy": "Geography",
  "type": "geo",
  "annotations": {}
}
```

Estructura

Una jerarquía es un tipo de categorización aplicada a las variables. Es útil si una variable presenta más de una forma de agrupar los distintos elementos que la componen (como la Geografía, que puede agruparse en sus Divisiones Políticas, en Áreas Metropolitanas, o bajo el Sistema Urbano Nacional).

```
interface Hierarchy {
  annotations: Annotations;
  name: string;
  levels: Level[];
}
```

```
$ echo $SCHEMA | \
jq '.cubes[0].dimensions[0].hierarchies[0]'

{
  "name": "Geography",
  "levels": [
    {
      "name": "Nation",
      "properties": [...],
      "annotations": {},
      "unique_name": null
    },
    {
      "name": "State",
      "properties": [...],
      "annotations": {},
      "unique_name": null
    },
    {
      "name": "Municipality",
      "properties": [...],
      "annotations": {},
      "unique_name": null
    }
  ],
  "annotations": {}
}
```

Estructura

Una jerarquía se compone de distintos niveles de agrupación.

```
interface Level {
  annotations: Annotations;
  name: string;
  properties: Property[];
  unique_name: string;
}
```

La abstracción obliga a que un nivel más bajo sólo pertenezca a un (1) nivel más alto. Una *Nación* se compone de varios *Estados*, y éstos a su vez de varios *Municipios*, pero un *Municipio* no puede pertenecer a más de un *Estado*, ni un *Estado* a más de una *Nación*.

```
$ echo $SCHEMA | \
jq '.cubes[0].dimensions[0].hierarchies[0]'

{
  "name": "Geography",
  "levels": [
    [...],
    {
      "name": "State",
      "properties": [
        {
          "name": "State ISO3",
          "caption_set": null,
          "annotations": {},
          "unique_name": null
        },
        {
          "name": "CVE State",
          "caption_set": null,
          "annotations": {},
          "unique_name": null
        },
        {
          "name": "State ES",
          "caption_set": null,
          "annotations": {},
          "unique_name": null
        },
      ],
      "annotations": {},
      "unique_name": null
    },
    [...]
  ],
  "annotations": {}
}
```

Estructura

En ocasiones, los *miembros* de un nivel pueden tener metainformación que es útil obtener al momento de realizar una consulta. Esta información se denomina propiedad.

```
interface Property {
  annotations: Annotations;
  name: string;
  caption_set?: string;
  unique_name?: string;
}
```

Como muestra el ejemplo, un *Estado* puede tener asociado además de su nombre original, un nombre distinto en otro idioma, un código ISO, u otra información.

Construcción de consultas

URL base de tesseract:
`https://api.datamexico.org/tesseract/`

Endpoint del esquema general:
`https://api.datamexico.org/tesseract/cubes`

Endpoint del esquema para un cubo específico: (en este ejemplo: `anuies_enrollment`)
`https://api.datamexico.org/tesseract/cubes/anuies_enrollment`

Endpoint para obtener el listado de miembros de un nivel:
`https://api.datamexico.org/tesseract/cubes/anuies_enrollment/members.extension`

Endpoint de consulta de datos para dicho cubo:
`https://api.datamexico.org/tesseract/cubes/anuies_enrollment/aggregate.extension`

La extensión al final establece el formato de los datos que retorna el servidor: jsonarrays, jsonrecords, csv.
Es opcional, si no se establece, se devuelve csv por defecto.

Construcción de consultas

```
const params = {
  drilldowns[]: ["Geography.Geography.Municipality", "Sex.Sex.Sex", "Year.Year.Year"],
    measures[]: ["Students"],
       parents: false,
        sparse: false
}
```

Los parámetros de la consulta se envían en los Search Params de la URL:

```
/aggregate.jsonrecords?drilldowns%5B%5D=Geography.Geography.Municipality&drilldowns%5B%5D=S
                       ex.Sex.Sex&drilldowns%5B%5D=Year.Year.Year&measures%5B%5D=Students&p
                       arents=false&sparse=false
```

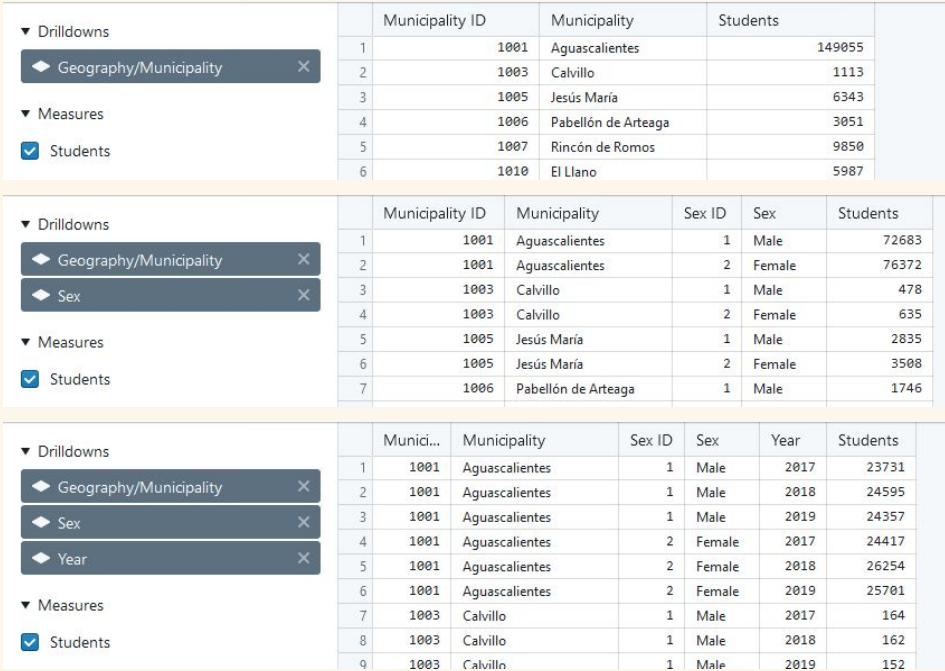
Su lenguaje de programación favorito ya debería tener una función o librería para hacer esta conversión. Al final de la presentación se verán ejemplos en Javascript, Python y Rstats.

Construcción de consultas

Para realizar una consulta, la información mínima que se requiere es el cubo de interés, al menos un nivel de profundidad (`drilldowns[]`) y al menos una métrica (`measures[]`). Sólo puede usarse un nivel de cada dimensión en la misma consulta.

Al momento de realizar una consulta, el cubo será dividido en los niveles que se especifiquen, y los grupos que queden serán agregados y calculados según la configuración de la respectiva métrica.

```
drilldowns[]: [
  "Dimension.Hierarchy.Level",
  "Dimension.Level"
],
measures[]: ["Students"]
```



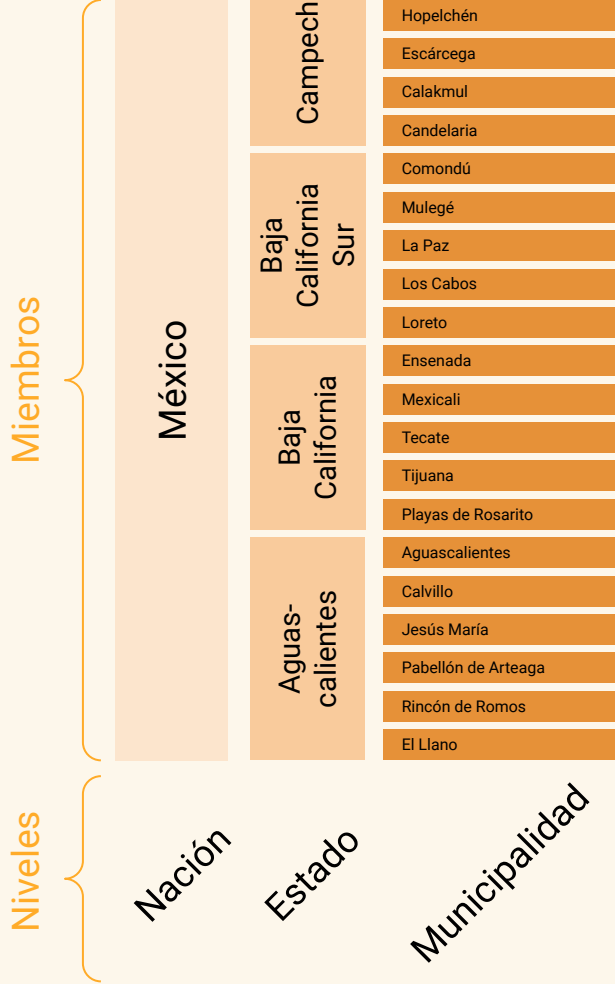
Construcción de consultas

Los *miembros* (`members`) de un *nivel* son el conjunto de valores que pueden tomar las variables asociadas a las mediciones dentro de alguno de los *niveles* de una *dimensión*.

El listado de *miembros* para un *nivel* se obtiene realizando otra petición al servidor, a la URL

```
https://api.datamexico.org/tesseract/cubes/cube_name/
  members.jsonrecords?level=Dimension.Hierarchy.Level
```

```
$ curl .../anuies_enrollment/members.jsonrecords?level=Sex.Sex
{
  "data": [
    {"ID": 1, "Label": "Male"},
    {"ID": 2, "Label": "Female"}
  ]
}
```



Construcción de consultas

Una consulta puede restringirse a un cierto grupo específico de *miembros* de cualquier *nivel*. Llamamos a esta característica un *corte* (`cuts[]`). Una consulta para obtener datos de todos los municipios dentro del Estado de Puebla (`id: 21`), para los años 2017-2019 (`id: 2017, 2018 y 2019`), se armaría así:

```
cuts[]: [
  "${'Dimension'.'Hierarchy'.'Level'},${member_id}",
  "${'Dimension'.'Level'},${member_id},${member_id},..."
]
```

```
cuts[]: [
  "Geography.Geography.State.21",
  "Year.Year.2017,2018,2019"
]
```

Se debe tener cuidado al utilizar múltiples cortes, pues si se forman incorrectamente, se podrían retornar conjuntos vacíos como resultado.

```
drilldowns[]: ["Geography.Municipality"],
cuts[]: [
  "Geography.Municipality.1001",   //  1001 = Municipio de Aguascalientes
  "Geography.State.21"             // 21    = Estado de Puebla
]
```

Construcción de consultas

Adicionalmente a filtrar por miembros, es posible filtrar por valores resultantes de calcular las métricas. Esto simplemente se denomina filtro (`filters[]`).

Un filtro se compone de una métrica, y una a dos restricciones. Una restricción se compone de un operador de comparación y un valor numérico. Si hay dos restricciones, deben unirse con un operador de unión.

Como ejemplo, obtener sólo los datos de municipalidades cuyo número de estudiantes es mayor a 100, y menor o igual a 500:

```
// Filtro de 1 restricción:
"Measure.comp.995"

// Filtro de 2 restricciones:
"Measure.comp1.991.and.comp2.999"

// Operaciones de comparación
"lt": menor a   "lte": menor o igual a
"gt": mayor a   "gte": mayor o igual a
"eq": igual a   "neq": no igual a

// Uniones
"and": ambas    "or": alguna de las dos

// Ejemplo
{
  filters[]: [
    "Students.gt.100.and.lte.500"
  ]
}
```

Cálculos de crecimiento

Para realizar cálculos de crecimiento, Tesseract necesita 2 parámetros: un nivel de división temporal, y una métrica para calcular. Adicionalmente, cada uno debe usarse en drilldowns y measures.

```
{
  drilldowns[]: "${'Dimension'.'Hierarchy'.'Level'}",
    measures[]: "${Measure}",
        growth: "${'Dimension'.'Hierarchy'.'Level'},${Measure}"
}
```

Estos parámetros añaden dos nuevos valores a cada fila:

`"${Measure} Growth"`:

$$\frac{\text{Measure}_n - \text{Measure}_{n-1}}{\text{Measure}_{n-1}}$$

`"${Measure} Growth Value"`:

$$\text{Measure}_n - \text{Measure}_{n-1}$$

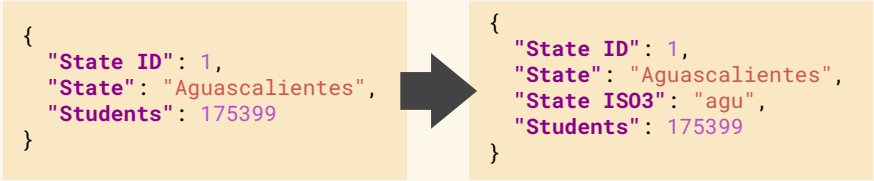

```
{
  "Municipality ID": 1001,
  "Municipality": "Aguascalientes",
  "Year": 2018,
  "Students": 50849,
  "Students Growth": 0.05609786491650744,
  "Students Growth Value": 2701
}
```

Parámetros adicionales

- `properties[]?: string[];`

Se utiliza para agregar a cada fila el valor de una propiedad de uno de los niveles usados como drilldown.

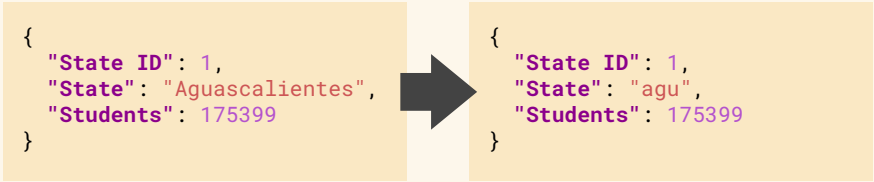
```
{
  drilldowns[]: ["Geography.State"],
  properties[]: ["Geography.State.State ISO3"]
}
```



- `captions[]?: string[];`

Se utiliza para reemplazar el valor de un drilldown, definido por el nivel, por el valor de una de las propiedades de dicho nivel.

```
{
  drilldowns[]: ["Geography.State"],
  captions[]: ["Geography.State.State ISO3"]
}
```



Parámetros adicionales

- `limit?: string = "${rows}" | "${offset},${rows}";`

Especifica la paginación de los resultados de la consulta. Las variables aquí representan números. Si una consulta retorna demasiados datos, puede dividirse en varias consultas, cada una con `rows` filas. Si se define `offset`, se retornarán `rows` elementos después de saltarse los primeros `offset`.

```
{
  limit: "4,10", // obtiene 10 elementos, partiendo desde el 5to en adelante
}
```

- `sort?: string = "${Measure}.asc" | "${Measure}.desc";`

Especifica un ordenamiento de los resultados de la consulta. Se debe usar el nombre de una métrica, o el término `"growth"` para los valores calculados para el crecimiento.

```
{
  sort: "Students.desc", // ordena por N° de estudiantes de mayor a menor
}
```

Parámetros adicionales

- `parents?: string = "true" | "false";`

Hace que el resultado de la consulta incluya, además de los niveles usados como drilldowns, la información de todos los niveles superiores a éstos.

	Municipality ID	Municipality	Students
1	1001	Aguascalientes	149055
2	1003	Calvillo	1113
3	1005	Jesús María	6343
4	1006	Pabellón de Arteaga	3051
5	1007	Rincón de Romos	9850
6	1010	El Llano	5987
7	2001	Ensenada	60427

	Nation ID	Nation	State ID	State	Municipality ID	Municipality	Students
1	mex	México	1	Aguascalientes	1001	Aguascalientes	149055
2	mex	México	1	Aguascalientes	1003	Calvillo	1113
3	mex	México	1	Aguascalientes	1005	Jesús María	6343
4	mex	México	1	Aguascalientes	1006	Pabellón de Arteaga	3051
5	mex	México	1	Aguascalientes	1007	Rincón de Romos	9850
6	mex	México	1	Aguascalientes	1010	El Llano	5987
7	mex	México	2	Baja California	2001	Ensenada	60427

- `sparse?: string = "true" | "false";`

Elimina del resultado los miembros que no tengan valores calculados. Esto es útil para disminuir el tamaño de descarga de sets de datos con agujeros de información (*sparse datasets*).

- `debug?: string = "true" | "false";`

Determina si la consulta se ejecuta en modo de depuración. En caso de error, el servidor registra información más detallada para solucionar problemas, y la respuesta hacia el cliente es más descriptiva.

Ejemplo en javascript


```
const baseUrl =
   "https://api.datamexico.org/tesseract/cubes/anuies_enrollment/aggregate.jsonrecords";
const params = {
  "drilldowns[]": ["Geography.Geography.Municipality", "Year.Year.Year"],
  "measures[]": ["Students"]
}

// Para generar la URL como string
const formUrlEncode = require("form-urlencoded"); // librería externa
const url = `${baseUrl}?${formUrlEncode(params)}`;

// Para descargar los datos directamente
const axios = require("axios"); // librería externa
const {data} = await axios.get(baseUrl, {params});
```

Ejemplo en Python

```
base_url =
   "https://api.datamexico.org/tesseract/cubes/anuies_enrollment/aggregate.jsonrecords"
payload = {
  "drilldowns[]": ["Geography.Geography.Municipality", "Year.Year.Year"],
  "measures[]": ["Students"]
}

# Para generar la URL como string
from urllib import parse

url = base_url + "?" + parse.urlencode(payload, True)

# Para descargar los datos directamente
import requests # librería externa

response = requests.get(base_url, params=payload)
data = response.json()
```

Ejemplo en R

```r
library(tidyverse); library(httr); library(jsonlite);

baseUrl <-
   "https://api.datamexico.org/tesseract/cubes/anuies_enrollment/aggregate.jsonrecords";
payload <- list(
  'drilldowns[]' = "Geography.Geography.Municipality",
  'drilldowns[]' = "Year.Year.Year",
  'measures[]' = "Students"
);

# Para generar la URL como string
url = httr::modify_url(baseUrl, query = payload);

# Para descargar los datos directamente
request <- httr::GET(baseUrl, query = payload);
response <- httr::content(request, as = "text", encoding = "UTF-8");
json <- fromJSON(response, flatten = TRUE);
```