ACONA Data Storage Concept

Requirements:

- Storing (semi)structured data centrally, that is easily accessible for analysis and display (single entry point)
- Performant: Querying data should be fast
- HIgh volume: We probably need to support big amount of data, like daily or hourly entries for a few hundred thousand urls
- Authentication should be possible

What will be saved?

Every hit/visit? Every hit/visit will not be available in tools like gsc. Also tools like matomo/ga do a better job saving those values. So we should save aggregated values (e.g by hour or day) in the data warehouse.

Data Lake: Saving raw data (if necessary) like crawl or lighthouse results. Tools like Matomo can be accessed directly. Data from services with api limits can be also saved periodically in the data lake, so we have access to all the data without limits.

Data Warehouse:
Saving structured content that can be easily accessed for analysis and display.

Key values:

-> Timestamp value
-> Url
-> Metric

Google search console (Performance Variables, Page variables)

API: https://developers.google.com/webmaster-tools/search-console-api-original/v3/?apix=true
https://developers.google.com/webmaster-tools/search-console-api-original/v3/how-tos/search_analytics

It's also possible to download all data per day:
https://developers.google.com/webmaster-tools/search-console-api-original/v3/how-tos/all-your-data

Minimum timeframe: 1 day
Dimensions; mobile, desktop, query?

```
request = {
     'startDate': flags.start_date,
     'endDate': flags.end_date,
     'dimensions': ['page'],
     'dimensionFilterGroups': [{
        'filters': [{
           'dimension': 'device',
           'expression': 'mobile'
        }]
     }],
    'rowLimit': 25000,
    'startRow': 0

 }
```

Keys	Clicks	Impressions	CTR	Position
https://www.example.com/21	10538.0	62639.0	0.168233847922	3.63031019014
https://www.example.com/65	9740.0	82375.0	0.118239757208	5.61003945372
https://www.example.com/15	9220.0	128101.0	0.0719744576545	5.32300294299
https://www.example.com/41	8859.0	426633.0	0.0207649197319	1.62309057199
https://www.example.com/53	8791.0	829679.0	0.0105956641062	14.4941887164
https://www.example.com/46	7390.0	82303.0	0.0897901656076	5.7723290767
https://www.example.com/27	7169.0	64013.0	0.111992876447	4.98709637105
https://www.example.com/80	6047.0	84233.0	0.0717889663196	4.10592048247
https://www.example.com/9	5886.0	59704.0	0.0985863593729	4.0897594801
https://www.example.com/8	5043.0	66869.0	0.0754161120998	4.57651527614

Url (string), Datetime (timestamp), Period (seconds), Metric (string), Value (integer), Client (string)

Example: https://www.example.com/21, 1597276800, 68400, OrganClicks, 10538, acolono

Links:
Example to write from json into a dataframe:
https://databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html

http://spark.apache.org/docs/latest/sql-programming-guide.html#parquet-files

Matomo (Success Variables)

https://matomo.org/docs/analytics-api/

Example result:
[{"2021-04-23":[],"2021-04-24":[],"2021-04-25":[],"2021-04-26":[],"2021-04-27":[],"2021-04-28":[],"2021-04-29":[],"2021-04-30":{"nb_uniq_visitors":1,"nb_users":0,"nb_visits":1,"nb_actions":3,"nb_visits_converted":1,"bounce_count":0,"sum_visit_length":102,"max_actions":3,"bounce_rate":"0\u00a0%","nb_actions_per_visit":3,"avg_time_on_site":102},"2021-05-01":[],"2021-05-02":[],"2021-05-03":[],"2021-05-04":[],"2021-05-05":[],"2021-05-06":[],"2021-05-07":{"nb_uniq_visitors":1,"nb_users":0,"nb_visits":1,"nb_actions":4,"nb_visits_converted":1,"bounce_count":0,"sum_visit_length":505,"max_actions":4,"bounce_rate":"0\u00a0%","nb_actions_per_visit":4,"avg_time_on_site":505},"2021-05-08":{"nb_uniq_visitors":1,"nb_users":0,"nb_visits":1,"nb_actions":1,"nb_visits_converted":0,"bounce_count":1,"sum_visit_length":0,"max_actions":1,"bounce_rate":"100\u00a0%","nb_actions_per_visit":1,"avg_time_on_site":0},"2021-05-09":[],"2021-05-10":[],"2021-05-11":[],"2021-05-12":[],"2021-05-13":[],"2021-05-14":[],"2021-05-15":[],"2021-05-16":[],"2021-05-17":[],"2021-05-18":[],"2021-05-19":[],"2021-05-20":[],"2021-05-21":[],"2021-05-22":[],"2021-05-23":[],"2021-05-24":[],"2021-05-25":[],"2021-05-26":[],"2021-05-27":[],"2021-05-28":[],"2021-05-29":[],"2021-05-30":[],"2021-05-31":[],"2021-06-01":[],"2021-06-02":[],"2021-06-03":[],"2021-06-04":[],"2021-06-05":[],"2021-06-06":[],"2021-06-07":[],"2021-06-08":[],"2021-06-09":[],"2021-06-10":[],"2021-06-11":[],"2021-06-12":[],"2021-06-13":[],"2021-06-14":[],"2021-06-15":[],"2021-06-16":[],"2021-06-17":[],"2021-06-18":[],"2021-06-19":[],"2021-06-20":[],"2021-06-21":[]}]

Matomo Serverlog Analytics (Performance Variables)

Log Analytics:
https://matomo.org/log-analytics/

Matomo Insights:
https://demo.matomo.org/?module=API&method=Insights.getInsightsOverview&idSite=62&period=day&date=today&format=JSON&token_auth=anonymous

ACONA SUCCESS VALUE

Table: acona_success

Url (string), Datetime (timestamp), Period (integer), Value Page Type 1 (Float), Value Page Type 2 (Float), Value Page Type 3 (Float), Value Page Type 4 (Float), Client (string)

Example: https://www.acona.app/metrics, 1597276800, 68400, 0.5, 0.4, 0.3, 0.7**, acolono**

For every defined page type a success value is saved. The client can define how these values are calculated by picking one or a combination of the available metrics (hits, visits, a matomo goal, scroll depth, …).
In the central settings page the client can also define a name for each page type.
In the CMS the client can set the page type for a specific url, so the right value is imported.

Max: 100, Low: 0

Note: For now we can just work with one page type! Period will be mostly one day (68400 seconds)

Rules

We have to store rules: Means min and max values for specific metrics, that can be compared with live data.

Types of rules:

1. We have generic default rules that are best practice.
2. Then we will have client specific rules, that are based on client specific analysis.

Saved in jsonlogic format: https://jsonlogic.com/ + metadata.

Can be maintained in ACONA Admin Backend and will be synched to the Data Warehouse. This would make it easy to store metrics and its goal values generally or by users, and make it accessible to external tools via (JSON-) API.

Data Tables

General infos:
https://blog.timescale.com/blog/time-series-data-why-and-how-to-use-a-relational-database-instead-of-nosql-d0cd6975e87c/
https://blog.timescale.com/blog/timescaledb-vs-6a696248104e/
https://docs.timescale.com/latest/api#create_hypertable

"The number of unique time series stored in the TSDB is called cardinality. "High cardinality" means high number of series. Prometheus tsdb storage is optimized for working with relatively low number of time series — see these slides from PromCon 2019. It could start working slowly or could use high amounts of RAM, CPU or disk IO during ingestion, querying or compaction when high number of time series is stored in it. High cardinality for Prometheus starts from a few millions of time series. Read this article about how to avoid high cardinality in Prometheus."
See
https://valyala.medium.com/prometheus-storage-technical-terms-for-humans-4ab4de6c3d48

Metric Naming in Prometheus: https://prometheus.io/docs/practices/naming/
The should have a suffix describing the unit. And a prefix describing the origin.

Example data structure: https://docs.timescale.com/latest/tutorials/tutorial-hello-timescale

It is possible to create and join hypertables (promscale) and regular postgres tables.

Metrics: Time Based integer values (hypertables)
Variables: Additional metainformation, can also store other formats

_d: Daily
_h: Hourly

```
CREATE DATABASE acona_data_warehouse;
\c acona_data_warehouse
CREATE EXTENSION IF NOT EXISTS timescaledb CASCADE;
```

What is the difference between Page Variables and Performance Metrics?
Performance Metrics are all of type integer and already aggregated (e.g. by day). They measure the performance of a specific url, e.g. how many times a user visited the page.

Page Variables define a specific state of the page for a given time. It can be integer (e.g. number of headings), holding text (e.g. the metadescription) or boolean (e.g. if breadcrumbs are present).

Performance Metrics

We want to have one table by metric (narrow table mode), not the wide table model: https://docs.timescale.com/latest/introduction/data-model

Metrics:
https://docs.google.com/spreadsheets/d/1UIwxvKds996FOvx372lN6WLteRD3rR3QGMClxihc7NA/edit?skip_itp2_check=true#gid=18291654

TimeScaleDB Narrow version:

create schema acona;

```
CREATE TABLE "metric_d_bounces"(
    url TEXT,
    date DATE NOT NULL,
    value INTEGER
```

```
    );
SELECT create_hypertable('metric_d_bounces', 'date',
create_default_indexes=>FALSE);
CREATE INDEX ON metric_d_bounces (url, date DESC);

CREATE TABLE "metric_d_page_views"(
    url TEXT,
    date DATE NOT NULL,
    value INTEGER
    );
SELECT create_hypertable("metric_d_page_views"', 'date',
create_default_indexes=>FALSE);
CREATE INDEX ON metric_d_page_views (url, date DESC);

CREATE TABLE "metric_d_visits"(
    url TEXT,
    date DATE NOT NULL,
    value INTEGER
    );
SELECT create_hypertable('metric_d_visits', 'date',
create_default_indexes=>FALSE);
CREATE INDEX ON metric_d_visits(url, date DESC);

CREATE TABLE "metric_d_unique_visits"(
    url TEXT,
    date DATE NOT NULL,
    value INTEGER
    );
SELECT create_hypertable('metric_d_unique_visits', 'date',
create_default_indexes=>FALSE);
CREATE INDEX ON metric_d_unique_visits(url, date DESC);

CREATE TABLE "metric_d_bounce_rate"(
    url TEXT,
    date DATE NOT NULL,
    value DECIMAL
    );
SELECT create_hypertable('metric_d_bounce_rate', 'date',
create_default_indexes=>FALSE);
CREATE INDEX ON metric_d_bounce_rate(url, date DESC);
```

An alternative with client_id, but better not do this because it adds a lot of timeseries (high cardinality problem).

{

"labels":{"__name__": "metric_d_bounces", “url”: “https://www.acolon.com/betterembed”, “client_id”: 1},
"samples":[
[1577836800000, 100],
[1577836801000, 99],
[1577836802000, 98],
...
]
}

Page Variables

Metrics: https://docs.google.com/spreadsheets/d/1UIwxvKds996FOvx372IN6WLteRD3rR3QGMClxihc7NA/edit?skip_itp2_check=true#gid=0

```
CREATE TABLE api.var_perf_ttfb_seconds(
    url TEXT NOT NULL,
    datetime TIMESTAMPTZ NOT NULL,
    value INTEGER
    );
SELECT create_hypertable('api.var_perf_ttfb_seconds', 'datetime',
create_default_indexes=>FALSE);
CREATE INDEX ON api.var_perf_ttfb_seconds (url, datetime DESC);

CREATE TABLE api.var_page_h1_number(
    url TEXT NOT NULL,
    datetime TIMESTAMPTZ NOT NULL,
    value INTEGER
    );
SELECT create_hypertable('api.var_page_h1_number', 'datetime',
create_default_indexes=>FALSE);
CREATE INDEX ON api.var_page_h1_number(url, datetime DESC);


CREATE TABLE api.var_page_metadescription(
    url TEXT NOT NULL,
    datetime TIMESTAMPTZ NOT NULL,
    value TEXT
    );
SELECT create_hypertable('api.var_page_metadescription', 'datetime',
create_default_indexes=>FALSE);
CREATE INDEX ON api.var_page_metadescription(url, datetime DESC);
```

FROM HUMANTEXT API
Example: https://humantxt.acona.app/?url=https://www.acolono.com/betterembed

```
CREATE TABLE api.var_page_word_count(
    url TEXT NOT NULL,
    datetime TIMESTAMPTZ NOT NULL,
    value INTEGER
    );
SELECT create_hypertable('api.var_page_word_count', 'datetime',
create_default_indexes=>FALSE);
CREATE INDEX ON api.var_page_word_count(url, datetime DESC);

CREATE TABLE api.var_page_content(
    url TEXT NOT NULL,
    datetime TIMESTAMPTZ NOT NULL,
    value TEXT
    );
SELECT create_hypertable('api.var_page_content', 'datetime',
create_default_indexes=>FALSE);
CREATE INDEX ON api.var_page_content(url, datetime DESC);

CREATE TABLE api.var_page_title(
    url TEXT NOT NULL,
    datetime TIMESTAMPTZ NOT NULL,
    value TEXT
    );
SELECT create_hypertable('api.var_page_title', 'datetime',
create_default_indexes=>FALSE);
CREATE INDEX ON api.var_page_title(url, datetime DESC);
```

Maybe also
(https://humantxt.acona.app/?url=https://www.acolono.com/betterembed&format=html):

```
CREATE TABLE api.var_page_content_html(
    url TEXT NOT NULL,
    datetime TIMESTAMPTZ NOT NULL,
    value TEXT
    );
SELECT create_hypertable('api.var_page_content_html', 'datetime',
create_default_indexes=>FALSE);
CREATE INDEX ON api.var_page_content_html(url, datetime DESC);
```

Success Metrics

The calculated ACONA Success Score for a specific URL and time.
Will be queried by CMS Integrations for specific URLs.
There are up to 3 types (page types).

```
CREATE TABLE "metric_success_score_type1_ratio"(
    url TEXT,
    date DATE NOT NULL,
    value INTEGER
    );
SELECT create_hypertable('metric_success_score_type1_ratio', 'date',
create_default_indexes=>FALSE);
CREATE INDEX ON metric_success_score_type1_ratio (url, date DESC);

CREATE TABLE "metric_success_score_type2_ratio"(
    url TEXT,
    date DATE NOT NULL,
    value INTEGER
    );
SELECT create_hypertable('metric_success_score_type2_ratio', 'date',
create_default_indexes=>FALSE);
CREATE INDEX ON metric_success_score_type2_ratio (url, date DESC);

CREATE TABLE "metric_success_score_type3_ratio"(
    url TEXT,
    date DATE NOT NULL,
    value INTEGER
    );
SELECT create_hypertable('metric_success_score_type3_ratio', 'date',
create_default_indexes=>FALSE);
CREATE INDEX ON metric_success_score_type3_ratio (url, date DESC);
```

Alternative: Check url pagetype already before writing and have only one table:

```
CREATE TABLE api.metric_success_score_ratio(
    url TEXT,
    date DATE NOT NULL,
    value decimal
    );
SELECT create_hypertable('api.metric_success_score_ratio', 'date',
create_default_indexes=>FALSE);
CREATE INDEX ON api.metric_success_score_ratio (url, date DESC);
```

Rules

Rules:
https://docs.google.com/spreadsheets/d/1UIwxvKds996FOvx372IN6WLteRD3rR3QGMCIxihc7NA/edit?skip_itp2_check=true#gid=813777100

Saved in jsonlogic format: https://jsonlogic.com/ + metadata.

Can be saved/maintained in ACONA Admin Backend.

For every rule “Title”, “Recommendation”, “Category”, “Relevance”, “Indication”, “More Info” can be queried from ACONA Admin Tool.

Rules Evaluation Metrics

~~Will be queried by CMS Integrations for specific URLs.~~
Will be queried by a tool that generates recommendations by combining rules metadata and evaluation results for a specific url and time.
Store for rule evaluations by time and url.

```
CREATE TABLE api.metric_rules_eval(
    url TEXT NOT NULL,
    date DATE NOT NULL,
    result BOOLEAN,
    rule_id VARCHAR(30) NOT NULL
    );
SELECT create_hypertable('api.metric_rules_eval', 'date',
create_default_indexes=>FALSE);
CREATE INDEX ON api.metric_rules_eval(rule_id, url, date DESC);
```

Example Query:
Get all rules that are evaluated TRUE for a specific url and date (day).
Get all recommendations for rules that are evaluated TRUE for a specific url, language and date (day).
Get all recommendations for rules that are evaluated FALSE for a specific url, language and date (day).

```
CREATE TABLE internal.acona_rules(
    rule_id VARCHAR(30) NOT NULL PRIMARY KEY,
    title_en TEXT,
```

```
    title_de TEXT,
    recommendation_en TEXT,
    recommendation_de TEXT,
    variable VARCHAR(30) NOT NULL,
    category TEXT,
    relevance INTEGER, /* 1-3 */
    indication VARCHAR(5) NOT NULL, /* green, yellow, red */
    condition JSON NOT NULL,
    more_de TEXT,
    more_en TEXT
);
```

Metrics

List of supported metrics and sources, description, type, unit, priority, machinename. Will be especially needed when we use the narrow storage option.

Can be saved/maintained in ACONA Admin Backend.

URLs

List of urls that should be analyzed.

Will be saved/maintained in ACONA Admin Backend (and synched to data warehouse).

Url (text), client_id (integer), domain (text), status (boolean), last (date), first (date), interval (text), pagetype (integer)

```
CREATE TABLE internal.urls(
    url TEXT NOT NULL PRIMARY KEY,
    user_id INTEGER NOT NULL,
    domain_id INTEGER,
    status BOOLEAN,
    intervall TEXT DEFAULT 'daily',
    pagetype INTEGER,
    first DATE,
    Last DATE
    );
```

Clients/Users

Can be saved/maintained in ACONA Admin Backend and synched to data warehouse.

Client id (integer), Status (boolean), Client name (text), E-Mail (text)

```
CREATE TABLE internal.users(
    user_id INTEGER NOT NULL PRIMARY KEY,
    user_name TEXT,
    status BOOLEAN,
    mail TEXT
);
```

Domains

We need a way to assign domains to users, as users only should have access to urls that are part of one of their domains. As theoretically more than one user should have access to a specific domain we have to create another table or use an array field.

```
CREATE TABLE internal.domains(
    domain_id INTEGER NOT NULL PRIMARY KEY,
    domain_name TEXT,
    users INTEGER[],
    synonyms TEXT[]
);
```

```
CREATE TABLE internal.domainusers(
    domain_id INTEGER NOT NULL,
    user_id INTEGER NOT NULL,
);
```

Variable calculation dates

Storing dates when a calculation has been done for a variable or table.

```
CREATE TABLE internal.var_calc_dates(
    variable VARCHAR(30) NOT NULL,
    date DATE NOT NULL,
    url TEXT NOT NULL
);
SELECT create_hypertable('internal.var_calc_dates', 'date',
create_default_indexes=>FALSE);
CREATE INDEX ON internal.var_calc_dates(variable, url);
```

API Views

Success Score by url and date

Arguments:
url: The url for that the success score should be queried
from_date: optional, default is today minus 7 days
to_date: optional: default is today.

```
CREATE OR REPLACE FUNCTION api.acona_success_scores(url TEXT, from_date DATE
DEFAULT now() - INTERVAL '7 days', to_date DATE DEFAULT now())
RETURNS table(date date, url text, value decimal) as $$
  SELECT
    date,
    url,
    value
  FROM api.metric_success_score_ratio s
  WHERE s.url = $1
  AND s.date >= $2
  AND s.date <= $3
  ORDER BY s.date DESC
$$ LANGUAGE SQL IMMUTABLE;
```

It can be used like

```
curl -G "localhost:3000/rpc/acona_success_scores"   -d url=https://www.acona.app/about;


$ curl -G "localhost:3000/rpc/acona_success_scores"   -d url=https://www.acona.app/about -d
from_date=2021-08-28;

curl -G "localhost:3000/rpc/acona_success_scores"   -d url=https://www.acona.app/about -d
from_date=2021-08-28 -d to_date=2021-08-29;
```

All Urls by domain with latest ACONA Success Score

Arguments:
domain: The domain for which all urls should be retrieved

```
CREATE OR REPLACE FUNCTION api.acona_urls_success(domain TEXT)
RETURNS table(url text, date date, value decimal) as $$
```

```
  SELECT
    urls.url,
    scores.date,
    scores.value
  FROM internal.urls urls
  LEFT JOIN api.metric_success_score_ratio scores
  ON (urls.url = scores.url)
  AND scores.date=(select max(date) from api.metric_success_score_ratio where url = urls.url)
  WHERE urls.domain_id = (
          SELECT domain_id
          FROM internal.domains domains
          WHERE domains.domain_name = $1
          );
 $$ LANGUAGE SQL IMMUTABLE
SECURITY DEFINER
    SET search_path = internal, pg_temp;
```

It can be used like

```
curl -G "localhost:3000/rpc/acona_urls_success"   -d domain=https://www.acona.app;
```

ACONA recommendations by url

Arguments:
url: The url for which recommendations should be retrieved
Date: Date for recommendations, default is today
Indication: red, yellow, green, default is all

```
CREATE OR REPLACE FUNCTION api.recommendations(url TEXT, date DATE DEFAULT
now(), indication TEXT DEFAULT 'red')
RETURNS table(indication text, title_en TEXT, title_de TEXT, recommendation_en  TEXT,
recommendation_de TEXT, date date, more_de TEXT, more_en TEXT, category TEXT,
relevance INTEGER) as $$
  SELECT
    rules.indication,
    rules.title_en,
    rules.title_de,
    rules.recommendation_en,
    rules.recommendation_de,
    eval.date,
    rules.more_de,
```

```
    rules.more_en,
    rules.category,
    rules.relevance
  FROM internal.acona_rules rules
  INNER JOIN api.metric_rules_eval eval
  ON (rules.rule_id = eval.rule_id
  AND eval.date=$2
  AND eval.url = $1)
  WHERE rules.indication = $3;
 $$ LANGUAGE SQL IMMUTABLE
SECURITY DEFINER
    SET search_path = internal, pg_temp;
```

It can be used like

```
curl -G "localhost:3000/rpc/recommendations"   -d url=https://www.acona.app/about -d
date=2021-09-06 -d indication=red;

curl -G "localhost:3000/rpc/recommendations"   -d url=https://www.acona.app/about -d
date=2021-09-06 -d indication=yellow;

curl -G "localhost:3000/rpc/recommendations"   -d url=https://www.acona.app/about -d
date=2021-09-06 -d indication=green;
```

Last ACONA recommendations by url

Arguments:
url: The url for which recommendations should be retrieved
Indication: red, yellow, green, default is all

```
CREATE OR REPLACE FUNCTION api.recommendations_last(url TEXT, indication TEXT
DEFAULT 'red')
RETURNS table(indication text, title_en TEXT, title_de TEXT, recommendation_en  TEXT,
recommendation_de TEXT, date date, more_de TEXT, more_en TEXT, category TEXT,
relevance INTEGER) as $$
  SELECT
    rules.indication,
    rules.title_en,
    rules.title_de,
    rules.recommendation_en,
    rules.recommendation_de,
    eval.date,
```

```
  rules.more_de,
  rules.more_en,
  rules.category,
  rules.relevance
 FROM internal.acona_rules rules
 INNER JOIN api.metric_rules_eval eval
 ON (rules.rule_id = eval.rule_id
 AND eval.date=(
        SELECT max(date)
        FROM internal.var_calc_dates calc_dates
        WHERE calc_dates.url = $1
        AND calc_dates.variable = 'metric_rules_eval'
        )
 AND eval.url = $1)
 WHERE rules.indication = $2;
$$ LANGUAGE SQL IMMUTABLE
SECURITY DEFINER
   SET search_path = internal, pg_temp;
```

It can be used like

curl -G "localhost:3000/rpc/recommendations_last" -d url=https://www.acona.app/about -d indication=red;

curl -G "localhost:3000/rpc/recommendations_last" -d url=https://www.acona.app/about -d indication=yellow;

curl -G "localhost:3000/rpc/recommendations_last" -d url=https://www.acona.app/about -d indication=green;