Integrum ESG ERD

An Entity Relationship Diagram to show a visual representation of how companies on the Integrum ESG Platform interact with business activities, metrics and sub-metrics for a metrics database built from the bulk APIs.

Introduction

As described in the API guide, for use cases where it is more efficient for the client to extract all Integrum ESG data into their own database, it is recommended to use bulk APIs like `company_cube_history` to retrieve everything at once.

The scores are structured in a "cube" format, with the level 1 data representing the sub-metrics and level 4 containing company-level data.

In the example below, we fetch `company_cube_history` for "Full Coverage" companies, and supplement it with proxy data calculated using sector and regional averages for the remaining companies.

After loading into the database (in this example SQLite), we ceate a view to provide scores for all companies.

To bring everything together, we retrieve all companies, metric names, and sector_metrics, which highlights the material metrics for sustainability assessments.

fullbulkdownload.sh

```
key="XXX"
today=`date -I`
companiesq="
query MyQuery {
  company_api_v2_with_proxies {
    id
    country
    isin
    name
    sector_id
    sector {
      name
    }
    business_activities {
      business_activity {
        name
      }
    }
    usd_revenue
    proxy
    icm
  }
}
"
proxyq="
query proxyQuery {
  proxy_cube_api(where: {profile_id: {_eq: -1}, yeardiff: {_eq: 0}}) {
    awareness_grade
    awareness_score
```

```
    country
    level
    level1
    level2
    level3
    metric_unit
    performance_grade
    performance_score
    performance_value
    scaled_performance_value
    sd
    sector
    total_grade
    total_score
    weighted_dimension_score
    yeardiff
  }
}
"
fcmq="
query FcmCubeQuery {
  company_cube_history(
    where: {record_date: {_eq: \\\"$today\\\"}}
  ) {
    company_id
    awareness_grade
    awareness_score
    company_id
    country
    financial_year
    level
    level1
    level2
    level3
    metric_unit
    name
    parent_sector
    performance_grade
    performance_score
    performance_value
    profile_id
    record_date
    scaled_performance_value
    sd
    sector
    sector_id
    total_grade
    total_score
    universe_id
    yeardiff
  }
}
"
smq="
query SectorMetricsQuery {
  sector_metrics {
    metric_id
    sector_id
  }
}
"
```

```
metricsq="
query MetricsQuery {
  metrics {
    id
    display_name
    dimension {
      name
      type
    }
  }
}
"
eq=`echo "$companiesq" | tr '\n' ' '`
curl -H "Authorization:Api-Key $key" -H "Content-Type: application/json" -H "Accept-Encoding: gzip" -
X POST -d "{\"query\": \"$eq\"}" https://dashboard.integrumesg.com/graphqlapi/v1/graphql >
COMPANIES.gz
eq=`echo "$fcmq" | tr '\n' ' '`
curl -H "Authorization:Api-Key $key" -H "Content-Type: application/json" -H "Accept-Encoding: gzip" -
X POST -d "{\"query\": \"$eq\"}" https://dashboard.integrumesg.com/graphqlapi/v1/graphql > FCM.gz
eq=`echo "$proxyq" | tr '\n' ' '`
curl -H "Authorization:Api-Key $key" -H "Content-Type: application/json" -H "Accept-Encoding: gzip" -
X POST -d "{\"query\": \"$eq\"}" https://dashboard.integrumesg.com/graphqlapi/v1/graphql > PROXY.gz
eq=`echo "$smq" | tr '\n' ' '`
curl -H "Authorization:Api-Key $key" -H "Content-Type: application/json" -X POST -d "{\"query\":
\"$eq\"}" https://dashboard.integrumesg.com/graphqlapi/v1/graphql > SM.json
eq=`echo "$metricsq" | tr '\n' ' '`
curl -H "Authorization:Api-Key $key" -H "Content-Type: application/json" -X POST -d "{\"query\":
\"$eq\"}" https://dashboard.integrumesg.com/graphqlapi/v1/graphql > METRICS.json
```

api_to_db.sh

```
#
# utility for converting a json array to a csv with the keys as columns
alias array_to_csv='jq -r '\''(map(keys) | add | unique) as $cols | map(. as $row | $cols |
map($row[.])) as $rows | $cols, $rows[] | @csv'\'''
# flatten out the nested elements to make companies csv
if [ "$1" != "-load" ]; then
    gzip -d < COMPANIES.gz| jq '[.data.company_api_v2_with_proxies[]| .["sector_name"] = .sector.name
| del(.sector) | del(.business_activities)]' | array_to_csv > companies.csv
    # normalize  the business_activities in a separate table
    gzip -d < COMPANIES.gz| jq '[ .data.company_api_v2_with_proxies[]|.id as
$id|  .business_activities[]| {company_id: $id, business_activity: .business_activity.name }] ' |
array_to_csv > business_activities.csv
    gzip -d < FCM.gz| jq .data.company_cube_history | array_to_csv > fcm.csv
    gzip -d < PROXY.gz| jq .data.proxy_cube_api | array_to_csv > proxy.csv
    jq .data.sector_metrics < sm.json| array_to_csv > sm.csv
    jq '[.data.metrics[]|{id: .id, level1: .display_name,  level2: .dimension.name,
level3: .dimension.type}]' < METRICS.json |  array_to_csv > metrics.csv
fi

sqlite3 myDatabase <<EOF
PRAGMA foreign_keys = ON;
create table metrics(id, level1 primary key, level2, level3);
.import --skip 1 metrics.csv metrics
create table companies (country, icm, id integer primary key, isin, name, proxy, sector_id,
sector_name, usd_revenue);
.mode csv
.import --skip 1 companies.csv companies
create table business_activities (business_activity, company_id integer
, foreign key(company_id) references companies(id)
);
```

```
.import --skip 1 business_activities.csv business_activities
create table fcm (awareness_grade, awareness_score, company_id, country, financial_year, level,
level1, level2, level3, metric_unit, name, parent_sector, performance_grade, performance_score,
performance_value, profile_id, record_date, scaled_performance_value, sd,sector, sector_id,
total_grade, total_score, universe_id, yeardiff
, foreign key(company_id) references companies(id)
, foreign key(level1) references metrics(level1)
);
.import --skip 1 fcm.csv fcm
create table proxy (awareness_grade, awareness_score, country, level, level1, level2, level3,
metric_unit, performance_grade, performance_score, performance_value, scaled_performance_value, sd,
sector, total_grade, total_score, weighted_dimension_score, yeardiff
, foreign key(level1) references metrics(level1)
);
.import --skip 1 proxy.csv proxy
create table sector_metrics(metric_id references metrics(id), sector_id
);
.import --skip 1 sm.csv sector_metrics
create view all_companies_cube as
select
    companies.country, id as company_id, isin, companies.name, proxy, companies.sector_id,
usd_revenue,
    awareness_grade, awareness_score, level, level1, level2, level3, metric_unit, performance_grade,
performance_score, performance_value, scaled_performance_value, sd, sector, total_grade, total_score
    from
 companies join proxy on companies.country = proxy.country and companies.sector_name = proxy.sector
where (companies.proxy = "true" or companies.icm = "true")
union all
select
    companies.country, id as company_id, isin, companies.name, proxy, companies.sector_id,
usd_revenue,
    awareness_grade, awareness_score, level, level1, level2, level3, metric_unit, performance_grade,
performance_score, performance_value, scaled_performance_value, sd, sector, total_grade, total_score
    from
 companies join fcm on companies.id = fcm.company_id  where (companies.proxy = "false" and
companies.icm = "false");
EOF
```

ERD for the resulting database