Package ‘geomultistar’

July 29, 2024

Type Package

Title Multidimensional Queries Enriched with Geographic Data

Version 1.2.2

Description Multidimensional systems allow complex queries to be carried out in an easy way. The geographical dimension, together with the temporal dimension, plays a fundamental role in multidimensional systems. Through this package, vector geographic data layers can be associated to the attributes of geographic dimensions, so that the results of multidimensional queries can be obtained directly as vector layers. The multidimensional structures on which we can define the queries can be created from a flat table or imported directly using functions from this package.

License MIT + file LICENSE


BugReports https://github.com/josesamos/geomultistar/issues

Depends R (>= 2.10)

Imports dplyr, generics, purrr, rlang, RSQLite, sf, tibble, tidyr, tidyselect

Suggests knitr, pander, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Jose Samos [aut, cre] (<https://orcid.org/0000-0002-4457-3439>), Universidad de Granada [cph]

Maintainer Jose Samos <jsamos@ugr.es>

Repository CRAN

Date/Publication 2024-07-29 05:20:02 UTC
## Contents

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_dimension</td>
<td>2</td>
</tr>
<tr>
<td>add_facts</td>
<td>4</td>
</tr>
<tr>
<td>define_geoattribute</td>
<td>6</td>
</tr>
<tr>
<td>dimensional_query</td>
<td>8</td>
</tr>
<tr>
<td>filter_dimension</td>
<td>9</td>
</tr>
<tr>
<td>geomultistar</td>
<td>10</td>
</tr>
<tr>
<td>get_empty_geoinstances</td>
<td>10</td>
</tr>
<tr>
<td>mrs_age_test</td>
<td>11</td>
</tr>
<tr>
<td>mrs_fact_age</td>
<td>12</td>
</tr>
<tr>
<td>mrs_fact_cause</td>
<td>12</td>
</tr>
<tr>
<td>mrs_when</td>
<td>13</td>
</tr>
<tr>
<td>mrs_where</td>
<td>13</td>
</tr>
<tr>
<td>mrs_who</td>
<td>14</td>
</tr>
<tr>
<td>ms_mrs</td>
<td>14</td>
</tr>
<tr>
<td>ms_mrs_test</td>
<td>15</td>
</tr>
<tr>
<td>multistar</td>
<td>15</td>
</tr>
<tr>
<td>multistar_as_flat_table</td>
<td>16</td>
</tr>
<tr>
<td>relate_dimension</td>
<td>17</td>
</tr>
<tr>
<td>run_geoquery</td>
<td>18</td>
</tr>
<tr>
<td>run_query</td>
<td>20</td>
</tr>
<tr>
<td>save_as_geopackage</td>
<td>21</td>
</tr>
<tr>
<td>select_dimension</td>
<td>23</td>
</tr>
<tr>
<td>select_fact</td>
<td>24</td>
</tr>
<tr>
<td>st_mrs_age_test</td>
<td>25</td>
</tr>
<tr>
<td>uk_london_boroughs</td>
<td>25</td>
</tr>
<tr>
<td>usa_cities</td>
<td>26</td>
</tr>
<tr>
<td>usa_counties</td>
<td>26</td>
</tr>
<tr>
<td>usa_divisions</td>
<td>27</td>
</tr>
<tr>
<td>usa_nation</td>
<td>27</td>
</tr>
<tr>
<td>usa_regions</td>
<td>28</td>
</tr>
<tr>
<td>usa_states</td>
<td>28</td>
</tr>
</tbody>
</table>

## Index

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_dimension</td>
<td>27</td>
</tr>
<tr>
<td>add_dimension</td>
<td>29</td>
</tr>
</tbody>
</table>

## Description

To add a dimension table to a multistar object, we must indicate the name that we give to the dimension, the tibble that contains the data and the name of the attribute corresponding to the table primary key.
add_dimension

Usage

```r
add_dimension(
  ms,
  dimension_name = NULL,
  dimension_table = NULL,
  dimension_key = NULL,
  fact_name = NULL,
  fact_key = NULL,
  key_as_data = FALSE
)
```

```r
## S3 method for class 'multistar'
add_dimension(
  ms,
  dimension_name = NULL,
  dimension_table = NULL,
  dimension_key = NULL,
  fact_name = NULL,
  fact_key = NULL,
  key_as_data = FALSE
)
```

Arguments

- **ms**: A `multistar` object.
- **dimension_name**: A string, name of dimension table.
- **dimension_table**: A tibble, dimension table.
- **dimension_key**: A string, name of the dimension primary key.
- **fact_name**: A string, name of fact table.
- **fact_key**: A string, name of the dimension foreign key.
- **key_as_data**: A boolean, define the primary key as an attribute of the dimension accessible in queries?

Details

We cannot add a dimension without defining a correspondence with one of the `multistar`'s fact tables. We have to define the name of the fact table and the name of its foreign key. The referential integrity of the instances of the facts is checked.

The attribute that is used as the primary key will no longer be accessible for queries (its function is considered to be exclusively related to facts). If you want to use it for queries, it must be explicitly indicated by the boolean parameter `key_as_data`.

Value

A `multistar`. 
add_facts

See Also

Other multistar functions: add_facts(), multistar(), relate_dimension()

Examples

```r
ms <- multistar() |> 
  add_facts(
    fact_name = "mrs_age",
    fact_table = mrs_fact_age,
    measures = "n_deaths",
    nrow_agg = "count"
  ) |> 
  add_facts(
    fact_name = "mrs_cause",
    fact_table = mrs_fact_cause,
    measures = c("pneumonia_and_influenza_deaths", "other_deaths"),
    nrow_agg = "nrow_agg"
  ) |> 
  add_dimension(
    dimension_name = "where",
    dimension_table = mrs_where,
    dimension_key = "where_pk",
    fact_name = "mrs_age",
    fact_key = "where_fk"
  ) |> 
  add_dimension(
    dimension_name = "when",
    dimension_table = mrs_when,
    dimension_key = "when_pk",
    fact_name = "mrs_age",
    fact_key = "when_fk",
    key_as_data = TRUE
  ) |> 
  add_dimension(
    dimension_name = "who",
    dimension_table = mrs_who,
    dimension_key = "who_pk",
    fact_name = "mrs_age",
    fact_key = "who_fk"
  )
```

add_facts  Add a fact table to a multistar

Description

To add a fact table to a multistar object, we must indicate the name that we give to the facts, the tibble that contains the data and a vector of attribute names corresponding to the measures.
**add_facts**

**Usage**

```r
add_facts(
  ms,
  fact_name = NULL,
  fact_table = NULL,
  measures = NULL,
  agg_functions = NULL,
  nrow_agg = "nrow_agg"
)
```

```r
## S3 method for class 'multistar'
add_facts(
  ms,
  fact_name = NULL,
  fact_table = NULL,
  measures = NULL,
  agg_functions = NULL,
  nrow_agg = "nrow_agg"
)
```

**Arguments**

- **ms**
  A multistar object.

- **fact_name**
  A string, name of fact table.

- **fact_table**
  A tibble, fact table.

- **measures**
  A vector of measure names.

- **agg_functions**
  A vector of aggregation function names. If none is indicated, the default is SUM. Additionally they can be MAX or MIN.

- **nrow_agg**
  A string, measurement name for the number of rows aggregated. If it does not exist, it is added to the table.

**Details**

Associated with each measurement, an aggregation function is required, which by default is SUM. It that can be SUM, MAX or MIN. Mean is not considered among the possible aggregation functions: The reason is that calculating the mean by considering subsets of data does not necessarily yield the mean of the total data.

An additional measurement, `nrow_agg`, corresponding to the number of aggregated rows is always added which, together with SUM, allows us to obtain the mean if needed. As the value of this parameter, you can specify an attribute of the table or the name that you want to assign to it (if it does not exist, it is added to the table).

**Value**

A multistar.
See Also

Other multistar functions: `add_dimension()`, `multistar()`, `relate_dimension()`

Examples

```r
e <- multistar() |> add_facts("fact_name = "mrs_age","fact_table = mrs_fact_age,
measures = "n_deaths",
row_agg = "count"
|>
add_facts("fact_name = "mrs_cause", 
         fact_table = mrs_fact_cause, 
         measures = c("pneumonia_and_influenza_deaths", "other_deaths"), 
         row_agg = "row_agg"
)
```

---

**define_geoattribute**  
*Define geographic attributes*

### Description

Defines a geographic attributes in two possible ways: Associates the instances of attributes of the geographic dimension with the instances of a geographic layer or defines it from the geometry of another previously defined geographic attribute. Multiple attributes can be specified in the attribute parameter.

#### Usage

```r
define_geoattribute(
  gms, 
  dimension = NULL, 
  attribute = NULL, 
  additional_attributes = NULL, 
  from_layer = NULL, 
  by = NULL, 
  from_attribute = NULL
)
```

```r
## S3 method for class 'geomultistar'
define_geoattribute(
  gms, 
  dimension = NULL, 
  attribute = NULL, 
  additional_attributes = NULL, 
```
define_geoattribute

from_layer = NULL,
by = NULL,
from_attribute = NULL
)

Arguments

gms A geomultistar object.
dimension A string, dimension name.
attribute A vector, attribute names.
additional_attributes A vector, attribute names.
from_layer A sf object.
by a vector of correspondence of attributes of the dimension with the sf layer structure.
from_attribute A string, attribute name.

Details

If defined from a layer (from_layer parameter), additionally the attributes used for the join between the tables (dimension and layer tables) must be indicated (by parameter).

If defined from another attribute, it should have a finer granularity, to obtain the result by grouping its instances.

If no value is indicated in the attribute parameter, it is defined for all those attributes of the dimension that do not have any previous definition, they are obtained from the attribute indicated in the from_attribute parameter.

Value

A geomultistar object.

See Also

Other geo functions: geomultistar(), get_empty_geoinstances(), run_geoquery()

Examples

gms <- geomultistar(ms = ms_mrs, geodimension = "where") |> 
define_geoattribute(
  attribute = "city",
  from_layer = usa_cities,
  by = c("city" = "city", "state" = "state")
)

gms <- gms |> 
define_geoattribute(attribute = c("region", "all_where"),
  from_attribute = "city"
dimensional_query

gms <- gms |>  
define_geoattribute(from_attribute = "city")

```r
gms <- gms |>  
define_geoattribute(attribute = "all_where",  
  from_layer = usa_nation)
```

dimensional_query dimensional_query S3 class

Description

An empty dimensional_query object is created where you can select fact measures, dimension attributes and filter dimension rows.

Usage

dimensional_query(ms = NULL)

Arguments

ms  
A multistar object.

Value

A dimensional_query object.

See Also

Other query functions: `filter_dimension()`, `run_query()`, `select_dimension()`, `select_fact()`

Examples

```r
# ms_mrs <- ct_mrs |>  
# constellation_as_multistar()  

dq <- dimensional_query(ms_mrs)
```
Description

Allows you to define selection conditions for dimension rows.

Usage

```r
filter_dimension(dq, name = NULL, ...)
```

## S3 method for class 'dimensional_query'

```r
filter_dimension(dq, name = NULL, ...)
```

Arguments

- `dq`: A `dimensional_query` object.
- `name`: A string, name of the dimension.
- `...`: Conditions, defined in exactly the same way as in `dplyr::filter`.

Details

Conditions can be defined on any attribute of the dimension (not only on attributes selected in the query for the dimension). The selection is made based on the function `dplyr::filter`. Conditions are defined in exactly the same way as in that function.

Value

A `dimensional_query` object.

See Also

Other query functions: `dimensional_query()`, `run_query()`, `select_dimension()`, `select_fact()`

Examples

```r
dq <- dimensional_query(ms_mrs) |>
  filter_dimension(name = "when", when_happened_week <= "03") |>
  filter_dimension(name = "where", city == "Boston")
```
### geomultistar

**Description**

A `geomultistar` object is created. Dimensions that contain geographic information are indicated.

**Usage**

```r
geomultistar(ms = NULL, geodimension = NULL)
```

**Arguments**

- `ms` A multistar structure.
- `geodimension` A vector of dimension names.

**Value**

A `geomultistar` object.

**See Also**

Other geo functions: `define_geoattribute()`, `get_empty_geoinstances()`, `run_geoquery()`

**Examples**

```r
# gms <- geomultistar(ms = ms_mrs, geodimension = "where")
```

---

### get_empty_geoinstances

**Description**

Get empty instances of a geographic attribute

**Usage**

```r
get_empty_geoinstances(gms, dimension = NULL, attribute = NULL)
```

**Examples**

```r
# gms <- get_empty_geoinstances(gms, dimension = NULL, attribute = NULL)
```
Arguments

gms  A geomultistar object.
dimension  A string, dimension name.
attribute  A string, attribute name.

Value

A sf object.

See Also

Other geo functions: define_geoattribute(), geomultistar(), run_geoquery()

Examples

gms <- geomultistar(ms = ms_mrs, geodimension = "where") |>  
    define_geoattribute(  
        attribute = "city",  
        from_layer = usa_cities,  
        by = c("city" = "city", "state" = "state")  
    )  

    empty <- gms |>  
        get_empty_geoinstances(attribute = "city")

mrs_age_test  Mortality Reporting System by Age Test

Description

Selection of data from the 2 Cities Mortality Reporting System by age group, for the first 3 weeks of 1962.

Usage

mrs_age_test

Format

A tibble.

Details

The original dataset begins in 1962. For each week, in 122 US cities, mortality figures by age group and cause, considered separately, are included (i.e., the combination of age group and cause is not included). In the cause, only a distinction is made between pneumonia or influenza and others. Two additional dates have been generated, which were not present in the original dataset.
Source


<table>
<thead>
<tr>
<th>mrs_fact_age</th>
<th>Fact age</th>
</tr>
</thead>
</table>

Description

Fact age table of the Mortality Reporting System. Defined from ms_mrs. Foreign keys have been renamed, only a when dimension has been considered, the type for the when dimension has been changed.

Usage

mrs_fact_age

Format

A tibble.

Source

https://CRAN.R-project.org/package=starschemar

<table>
<thead>
<tr>
<th>mrs_fact_cause</th>
<th>Fact cause</th>
</tr>
</thead>
</table>

Description

Fact cause table of the Mortality Reporting System. Defined from ms_mrs. Foreign keys have been renamed, only a when dimension has been considered, the type for the when dimension has been changed.

Usage

mrs_fact_cause

Format

A tibble.

Source

https://CRAN.R-project.org/package=starschemar
**mrs_when**

<table>
<thead>
<tr>
<th>mrs_when</th>
<th>Dimension when</th>
</tr>
</thead>
</table>

**Description**

*When* dimension table of the Mortality Reporting System. Defined from `ms_mrs`. The primary key has been renamed and its type has been changed. The other attributes have also been renamed.

**Usage**

```
mrs_when
```

**Format**

A tibble.

**Source**

[https://CRAN.R-project.org/package=starschemar](https://CRAN.R-project.org/package=starschemar)

---

**mrs_where**

<table>
<thead>
<tr>
<th>mrs_where</th>
<th>Dimension where</th>
</tr>
</thead>
</table>

**Description**

*Where* dimension table of the Mortality Reporting System. Defined from `ms_mrs`. The primary key has been renamed.

**Usage**

```
mrs_where
```

**Format**

A tibble.

**Source**

[https://CRAN.R-project.org/package=starschemar](https://CRAN.R-project.org/package=starschemar)
<table>
<thead>
<tr>
<th>mrs_who</th>
<th>Dimension who</th>
</tr>
</thead>
</table>

**Description**

*Who* dimension table of the Mortality Reporting System. Defined from *ms_mrs*. The primary key has been renamed.

**Usage**

`mrs_who`

**Format**

A tibble.

**Source**

[https://CRAN.R-project.org/package=starschemar](https://CRAN.R-project.org/package=starschemar)

<table>
<thead>
<tr>
<th>ms_mrs</th>
<th>Multistar for Mortality Reporting System</th>
</tr>
</thead>
</table>

**Description**

Multistar for the Mortality Reporting System considering age and cause classification.

**Usage**

`ms_mrs`

**Format**

A multistar object.

**Source**

[https://CRAN.R-project.org/package=starschemar](https://CRAN.R-project.org/package=starschemar)
Description

Multistar for the Mortality Reporting System considering age and cause classification data test.

Usage

ms_mrs_test

Format

A multistar object.

Source

https://CRAN.R-project.org/package=starschemar

Description

Creates an empty multistar object that allows you to import fact and dimension tables.

Usage

multistar()

Value

A multistar object.

See Also

Other multistar functions: add_dimension(), add_facts(), relate_dimension()

Examples

ms <- multistar()
multistar_as_flat_table

*Export a multistar as a flat table*

**Description**

We can obtain a flat table, implemented using a tibble, from a multistar (which can be the result of a query). If it only has one fact table, it is not necessary to provide its name.

**Usage**

```r
multistar_as_flat_table(ms, fact = NULL)
```

## S3 method for class 'multistar'

```r
multistar_as_flat_table(ms, fact = NULL)
```

**Arguments**

- `ms` A multistar object.
- `fact` A string, name of the fact.

**Value**

A tibble.

**Examples**

```r
ft <- ms_mrs |>
  multistar_as_flat_table(fact = "mrs_age")

ms <- dimensional_query(ms_mrs) |>
  select_dimension(name = "where", attributes = c("city", "state")) |>
  select_dimension(name = "when", attributes = c("when_happened_year")) |>
  select_fact(name = "mrs_age", measures = c("n_deaths")) |>
  select_fact(
    name = "mrs_cause", measures = c("pneumonia_and_influenza_deaths", "other_deaths")
  ) |>
  filter_dimension(name = "when", when_happened_week <= "03") |>
  filter_dimension(name = "where", city == "Boston") |>
  run_query()

ft <- ms |>
  multistar_as_flat_table()
```
relate_dimension

Relate a dimension table to a fact table in a multistar

Description

Adding a dimension to a multistar can only relate to a fact table. You can then relate to other fact tables in the multistar using this function. The name of the fact table and its foreign key must be indicated. The referential integrity of the instances of the facts is checked.

Usage

relate_dimension(ms, dimension_name = NULL, fact_name = NULL, fact_key = NULL)

## S3 method for class 'multistar'
relate_dimension(ms, dimension_name = NULL, fact_name = NULL, fact_key = NULL)

Arguments

ms A multistar object.
dimension_name A string, name of dimension table.
fact_name A string, name of fact table.
fact_key A string, name of the dimension foreign key.

Value

A multistar.

See Also

Other multistar functions: add_dimension(), add_facts(), multistar()

Examples

ms <- multistar() |>
  add_facts(
  fact_name = "mrs_age",
  fact_table = mrs_fact_age,
  measures = "n_deaths",
  nrow_agg = "count"
  ) |>
  add_facts(
  fact_name = "mrs_cause",
  fact_table = mrs_fact_cause,
  measures = c("pneumonia_and_influenza_deaths", "other_deaths"),
  nrow_agg = "nrow_agg"
  ) |>
  add_dimension(
  dimension_name = "where"},
run_geoquery

Get a geographic vector from a query

Description

After defining a query and geographic dimensions, run the query and select the geographic data associated with it to get a geographic data layer as the result.

Usage

run_geoquery(
    dq,
    unify_by_grain = TRUE,
    fact = NULL,
    dimension = NULL,
    attribute = NULL,
    wider = FALSE
)

## S3 method for class 'dimensional_query'
run_geoquery(

run_geoquery

dq,
unify_by_grain = TRUE,
fact = NULL,
dimension = NULL,
attribute = NULL,
wider = FALSE
)

Arguments

dq A dimensional_query object.
unify_by_grain A boolean, unify facts with the same grain.
fact A string, name of the fact.
dimension A string, name of the geographic dimension.
attribute A string, name of the geographic attribute to consider.
wider A boolean, avoid repeating geographic data.

Details

In the case of having several fact tables, as an option, we can indicate if we do not want to unify the facts in the case of having the same grain.

If the result only has one fact table, it is not necessary to provide its name. Nor is it necessary to indicate the name of the geographic dimension if there is only one available.

If no attribute is specified, the geographic attribute of the result with finer granularity is selected.

In geographic layers, geographic objects are not repeated. The tables are wide: for each object the rest of the attributes are defined as columns. By means of the parameter wider we can indicate that we want a result of this type.

Value

A sf object.

See Also

Other geo functions: define_geoattribute(), geomultistar(), get_empty_geoinstances()

Examples

gms <- geomultistar(ms = ms_mrs, geodimension = "where") |> define_geoattribute(
  attribute = "city",
  from_layer = usa_cities,
  by = c("city" = "city", "state" = "state")
) |> define_geoattribute(
  attribute = "state",
  from_layer = usa_states,
  by = c("state" = "state")
)
run_query

Once we have selected the facts, dimensions and defined the conditions on the instances, we can execute the query to obtain the result.

Usage

run_query(dq, unify_by_grain = TRUE)

## S3 method for class 'dimensional_query'
run_query(dq, unify_by_grain = TRUE)

Arguments

dq

A dimensional_query object.

unify_by_grain

A boolean, unify facts with the same grain.
Details

As an option, we can indicate if we do not want to unify the facts in the case of having the same grain.

Value

A dimensional_query object.

See Also

Other query functions: dimensional_query(), filter_dimension(), select_dimension(), select_fact()

Examples

```r
ms <- dimensional_query(ms_mrs) |
  select_dimension(name = "where",
                   attributes = c("city", "state")) |
  select_dimension(name = "when",
                   attributes = c("when_happened_year")) |
  select_fact(
    name = "mrs_age",
    measures = c("n_deaths"),
    agg_functions = c("MAX")
  ) |
  select_fact(
    name = "mrs_cause",
    measures = c("pneumonia_and_influenza_deaths", "other_deaths")
  ) |
  filter_dimension(name = "when", when_happened_week <= "03") |
  filter_dimension(name = "where", city == "Boston") |
  run_query()
```

save_as_geopackage

Save as geopackage

Description

Save the result of a geoquery in a geopackage. The result can be a layer in the form of a flat table or a list consisting of a layer and a description table of the variables.

Usage

```r
save_as_geopackage(sf, layer_name, file_name = NULL, filepath = NULL)
```
Arguments

- **sf**: A tibble or a list of tibble objects.
- **layer_name**: A string.
- **file_name**: A string.
- **filepath**: A string.

Value

A tibble or a list of tibble objects.

Examples

gms <- geomultistar(ms = ms_mrs, geodimension = "where") |>  
  define_geoattribute(  
    attribute = "city",  
    from_layer = usa_cities,  
    by = c("city" = "city", "state" = "state")  
  ) |>  
  define_geoattribute(  
    attribute = "state",  
    from_layer = usa_states,  
    by = c("state" = "state")  
  ) |>  
  define_geoattribute(attribute = "region",  
    from_attribute = "state")  
  define_geoattribute(attribute = "all_where",  
    from_layer = usa_nation)  

  gdq <- dimensional_query(gms) |>  
  select_dimension(name = "where",  
    attributes = c("state", "city")) |>  
  select_dimension(name = "when",  
    attributes = c("when_happened_year", "when_happened_week")) |>  
  select_fact(  
    name = "mrs_age",  
    measures = c("n_deaths")  
  ) |>  
  select_fact(name = "mrs_cause",  
    measures = c("pneumonia_and_influenza_deaths", "other_deaths")) |>  
  filter_dimension(name = "when", when_happened_week <= "03") |>  
  filter_dimension(name = "where", state == "MA")  

  sf <- gdq |>  
  run_geoquery(wider = TRUE)  

  save_as_geopackage(sf, "city", filepath = tempdir())
**select_dimension**

**Select dimension**

**Description**

To add a dimension in a `dimensional_query` object, we have to define its name and a subset of the dimension attributes. If only the name of the dimension is indicated, it is considered that all its attributes should be added.

**Usage**

```r
select_dimension(dq, name = NULL, attributes = NULL)
```

```r
## S3 method for class 'dimensional_query'
select_dimension(dq, name = NULL, attributes = NULL)
```

**Arguments**

- `dq`: A `dimensional_query` object.
- `name`: A string, name of the dimension.
- `attributes`: A vector of attribute names.

**Value**

A `dimensional_query` object.

**See Also**

Other query functions: `dimensional_query()`, `filter_dimension()`, `run_query()`, `select_fact()`

**Examples**

```r
dq <- dimensional_query(ms_mrs) |> select_dimension(name = "where", attributes = c("city", "state")) |> select_dimension(name = "when")
```
select_fact  Select fact

Description

To define the fact to be consulted, its name is indicated, optionally, a vector of names of selected measures and another of aggregation functions are also indicated.

Usage

select_fact(dq, name = NULL, measures = NULL, agg_functions = NULL)

## S3 method for class 'dimensional_query'
select_fact(dq, name = NULL, measures = NULL, agg_functions = NULL)

Arguments

dq      A dimensional_query object.
name    A string, name of the fact.
measures A vector of measure names.
agg_functions A vector of aggregation function names. If none is indicated, those defined in the fact table are considered.

Details

If the name of any measure is not indicated, only the one corresponding to the number of aggregated rows is included, which is always included.

If no aggregation function is included, those defined for the measures are considered.

Value

A dimensional_query object.

See Also

Other query functions: dimensional_query(), filter_dimension(), run_query(), select_dimension()

Examples

dq <- dimensional_query(ms_mrs) |> 
  select_fact( 
    name = "mrs_age", 
    measures = c("n_deaths"), 
    agg_functions = c("MAX")
  )

dq <- dimensional_query(ms_mrs) |>
select_fact(name = "mrs_age",
        measures = c("n_deaths"))

dq <- dimensional_query(ms_mrs) |>
  select_fact(name = "mrs_age")

---

### st_mrs_age_test

**Star Schema for Mortality Reporting System by Age Test**

**Description**

Star Schema for the Mortality Reporting System considering the age classification data test.

**Usage**

`st_mrs_age_test`

**Format**

A `star_schema` object.

**Source**

https://CRAN.R-project.org/package=starschema

---

### uk_london_boroughs

**UK London Boroughs**

**Description**

From the original dataset, some fields have been selected and renamed.

**Usage**

`uk_london_boroughs`

**Format**

A `sf`.

**Details**

Since not so much detail is needed, the geometry has been simplified 20 m.

**Source**

https://data.london.gov.uk/dataset/statistical-gis-boundary-files-london
usa_cities  USA Cities, 2014

Description
From the original dataset, some fields have been selected and renamed, and only includes the Mortality Reporting System cities.

Usage
usa_cities

Format
A sf.

Source
https://earthworks.stanford.edu/catalog/stanford-bx729wr3020

usa_counties  USA Counties, 2018

Description
From the original dataset, some fields have been selected and renamed, and only includes the Mortality Reporting System counties.

Usage
usa_counties

Format
A sf.

Details
Some counties appear with the same repeated name within the same state, they are the following: Baltimore, MD; Richmond, VA; St. Louis, MO. Since they are accessed by name (county and state), those of the same name within the state have been grouped together.

Source
https://www2.census.gov/geo/tiger/GENZ2018/shp/cb_2018_us_county_20m.zip
usa_divisions  USA Divisions, 2018

Description
From the original dataset, some fields have been selected and renamed.

Usage
usa_divisions

Format
A sf.

Source
https://www2.census.gov/geo/tiger/GENZ2018/shp/cb_2018_us_division_20m.zip

usa_nation  USA Nation, 2018

Description
From the original dataset, some fields have been selected and renamed.

Usage
usa_nation

Format
A sf.

Source
https://www2.census.gov/geo/tiger/GENZ2018/shp/cb_2018_us_nation_20m.zip
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Description</th>
<th>Usage</th>
<th>Format</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>usa_regions</td>
<td>From the original dataset, some fields have been selected and renamed.</td>
<td>usa_regions</td>
<td>A sf.</td>
<td><a href="https://www2.census.gov/geo/tiger/GENZ2018/shp/cb_2018_us_region_20m.zip">https://www2.census.gov/geo/tiger/GENZ2018/shp/cb_2018_us_region_20m.zip</a></td>
</tr>
<tr>
<td>usa_states</td>
<td>From the original dataset, some fields have been selected and renamed, and only includes the Mortality Reporting System states.</td>
<td>usa_states</td>
<td>A sf.</td>
<td><a href="https://www2.census.gov/geo/tiger/GENZ2018/shp/cb_2018_us_state_20m.zip">https://www2.census.gov/geo/tiger/GENZ2018/shp/cb_2018_us_state_20m.zip</a></td>
</tr>
</tbody>
</table>
Index

* datasets
  mrs_age_test, 11
  mrs_fact_age, 12
  mrs_fact_cause, 12
  mrs_when, 13
  mrs_where, 13
  mrs_who, 14
  ms_mrs, 14
  ms_mrs_test, 15
  st_mrs_age_test, 25
  uk_london_boroughs, 25
  usa_cities, 26
  usa_counties, 26
  usa_divisions, 27
  usa_nation, 27
  usa_regions, 28
  usa_states, 28

* geo functions
  define_geoattribute, 6
  geomultistar, 10
  get_empty_geoinstances, 10
  run_geoquery, 18

* multistar functions
  add_dimension, 2
  add_facts, 4
  multistar, 15
  relate_dimension, 17

* query functions
  dimensional_query, 8
  filter_dimension, 9
  run_query, 20
  select_dimension, 23
  select_fact, 24

* results export functions
  multistar_as_flat_table, 16

add_dimension, 2, 6, 15, 17
add_facts, 4, 4, 15, 17

dimensional_query, 8, 9, 21, 23, 24
filter_dimension, 8, 9, 21, 23, 24
geomultistar, 7, 10, 11, 19
get_empty_geoinstances, 7, 10, 11, 19
mrs_age_test, 11
mrs_fact_age, 12
mrs_fact_cause, 12
mrs_when, 13
mrs_where, 13
mrs_who, 14
ms_mrs, 14
ms_mrs_test, 15
st_mrs_age_test, 25
multistar, 4, 6, 15, 17
multistar_as_flat_table, 16
relate_dimension, 4, 6, 15, 17
run_geoquery, 7, 10, 11, 18
run_query, 8, 9, 20, 23, 24
save_as_geopackage, 21
select_dimension, 8, 9, 21, 23, 24
select_fact, 8, 9, 21, 23, 24
st_mrs_age_test, 25
uk_london_boroughs, 25
usa_cities, 26
usa_counties, 26
usa_divisions, 27
usa_nation, 27
usa_regions, 28
usa_states, 28