Package: n2kupdate (via r-universe)

October 4, 2024

Title Auxiliary Functions to Update the n2kresult Database

Version 0.1.1

Date 2019-03-15

Description The functions are useful to store the results from https:// github.com/inbo/n2kanalysis into a PostgreSQL database created with https:// github.com/inbo/n2kresult.

Depends R (>= 3.2.0)

Imports assertthat, DBI, digest, dplyr, methods, purrr, rlang, RPostgreSQL, tibble, tidyr

Suggests aws.s3, n2kanalysis, optimx, testthat

Remotes inbo/n2kanalysis

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Collate 'character_df.R' 'class_n2kAnalysisVersion.R' 'connect_ut_db.R' 'import_S3_classes.R' 'sha1.R' 'store_analysis.R' 'store_analysis_dataset.R' 'store_analysis_version.R' 'store_anomaly.R' 'store_anomaly_type.R' 'store_datafield.R' 'store_datafield_type.R' 'store_dataset.R' 'store_datasource.R' 'store_datasource_parameter.R' 'store_datasource_type.R' 'store_language.R' 'store_location.R' 'store_location_group.R' 'store_location_group_location.R' 'store_model_set.R' 'store_model_type.R' 'store_n2kImport.R' 'store_n2kManifest.R' 'store_n2kModel.R' 'store_n2kResult.R' 'store_observation.R' 'store_parameter.R' 'store_scheme.R' 'store_source_species.R' 'store_source_species_species.R' 'store_species.R' 'store_species_group.R' 'store_species_group_species.R' 'store_status.R' 'truncate_public.R'

Repository https://inbo.r-universe.dev

RemoteUrl https://github.com/inbo/n2kupdate

RemoteRef HEAD

RemoteSha 3768ba50853b344f4a4f357334eabf5058e074aa

Contents

`character_df` *Convert all factors in a data.frame to characters*

Description

Convert all factors in a data.frame to characters

Usage

```
character_df(x, ...)
```

Arguments

`x`	object to be coerced or tested.
`...`	further arguments passed to or from other methods.

`connect_ut_db` *connect to the unit test database*

Description

connect to the unit test database

Usage

```
connect_ut_db(host = "localhost", dbname = "n2kunittest",
  user = "unittest_analysis", password = "unittest", port = 5432)
```

Arguments

`host`	the hostname of the database. Defaults to "localhost".
`dbname`	the name of the unit test database. Defaults to "n2kunittest".
`user`	the name of the unit test user. Defaults to "unittest_analysis".
`password`	the password for the user. Defaults to "unittest".
`port`	The port of host. Defaults to 5432.

`n2kAnalysisVersion-class`
The n2kAnalysisVersion class

Description

The n2kAnalysisVersion class

store_analysis — *store source species in the database*

Description

store source species in the database

Usage

```
store_analysis(analysis, model_set, analysis_version, analysis_relation,
  conn, hash, clean = TRUE)
```

Arguments

analysis	a data.frame with file_fingerprint, model_set_local_id, location_group, species_group, last_year, seed, analysis_version, analysis_date, status and status_fingerprint.
model_set	a data.frame with the model sets. Must have variables "local_id", "description", "first_year", "last_year" and "duration". The variable "long_description" is optional.
analysis_version	an n2kAnalysisVersion object. See get_analysis_version
analysis_relation	an optional data.frame with analysis and source_analysis. analysis contains the file_fingerprint of the current analysis. source_analysis contains the file_fingerprint of the parent analysis
conn	a DBIconnection
hash	the hash of the update session
clean	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

store_analysis_dataset — *store analysis and dataset in the database*

Description

store analysis and dataset in the database

Usage

```
store_analysis_dataset(analysis, model_set, analysis_version, dataset,
  analysis_dataset, clean = TRUE, hash, conn)
```

Arguments

`analysis`	a data.frame with file_fingerprint, model_set_local_id, location_group, species_group, last_year, seed, analysis_version, analysis_date, status and status_fingerprint.
`model_set`	a data.frame with the model sets. Must have variables "local_id", "description", "first_year", "last_year" and "duration". The variable "long_description" is optional.
`analysis_version`	an n2kAnalysisVersion object. See `get_analysis_version`
`dataset`	a data.frame with names fingerprint, filename, datasource and import_date
`analysis_dataset`	A `data.frame` linking the `file_fingerprint` from `analysis` to the `fingerprint` from `dataset`.
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.
`hash`	the hash of the update session
`conn`	a DBIconnection

`store_analysis_version` *Store the analysis version in the database*

Description

Store the analysis version in the database

Usage

```
store_analysis_version(analysis_version, hash, clean = TRUE, conn)
```

Arguments

`analysis_version`	an n2kAnalysisVersion object. See `get_analysis_version`
`hash`	the hash of the update session
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.
`conn`	a DBIconnection

`store_anomaly` *Store anomaly*

Description

Store anomaly

Usage

```
store_anomaly(anomaly, anomaly_type, parameter, hash, conn, clean = TRUE)
```

Arguments

`anomaly`	a data.frame with variables "anomaly_type_local_id", "datafield", "analyis" and "parameter_local_id".
`anomaly_type`	a data.frame with variables "local_id", "description" and "long_description". "long_description" is optional
`parameter`	a data.frame with parameters. Must contains the variables "description", "local_id", and "parent_parameter_local_id". Other variables are ignored.
`hash`	the hash of the update session
`conn`	a DBIconnection
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_anomaly_type` *Store anomaly types*

Description

Store anomaly types

Usage

```
store_anomaly_type(anomaly_type, hash, conn, clean = TRUE)
```

Arguments

`anomaly_type`	a data.frame with variables "local_id", "description" and "long_description". "long_description" is optional
`hash`	the hash of the update session
`conn`	a DBIconnection
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_datafield` *store a datafield in the database*

Description

store a datafield in the database

Usage

```
store_datafield(datafield, conn, hash, clean = TRUE)
```

Arguments

`datafield`	a data.frame with datafield metadata. Must contain the variables local_id, datasource, table_name, primary_key and datafield_type. Other variables are ignored.
`conn`	a DBIconnection
`hash`	the hash of the update session
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_datafield_type` *Store a vector of datafield types*

Description

Store a vector of datafield types

Usage

```
store_datafield_type(datafield_type, hash, conn, clean = TRUE)
```

Arguments

`datafield_type`	the vector with datafield types.
`hash`	the hash of the update session
`conn`	a DBIconnection
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

store_dataset *Store a dataset is the database*

Description

Store a dataset is the database

Usage

```
store_dataset(dataset, conn, clean = TRUE, hash)
```

Arguments

`dataset`	a data.frame with names fingerprint, filename, datasource and import_date
`conn`	a DBIconnection
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.
`hash`	the hash of the update session

store_datasource *store a datasource in the database*

Description

store a datasource in the database

Usage

```
store_datasource(datasource, conn, clean = TRUE, hash)
```

Arguments

`datasource`	a data.frame with datasource metadata
`conn`	a DBIconnection
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.
`hash`	the hash of the update session

Details

datasource must contain at least the variables description, datasource_type and connect_method.

`store_datasource_parameter` *Store a vector of datasource parameters*

Description

Store a vector of datasource parameters

Usage

```
store_datasource_parameter(datasource_parameter, hash, conn,
  clean = TRUE)
```

Arguments

`datasource_parameter`	the vector with datasource parameters.
`hash`	the hash of the update session
`conn`	a DBIconnection
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_datasource_type` *Store a vector of datasource types*

Description

Store a vector of datasource types

Usage

```
store_datasource_type(datasource_type, hash, conn, clean = TRUE)
```

Arguments

`datasource_type`	the vector with datasource types.
`hash`	the hash of the update session
`conn`	a DBIconnection
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_language` *Store language*

Description

Store language

Usage

```
store_language(language, hash, conn, clean = TRUE)
```

Arguments

`language`	the data.frame with language Must contains code and description. Other variables are ignored. code and description must have unique values.
`hash`	the hash of the update session
`conn`	a DBIconnection
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_location` *store locations in the database*

Description

store locations in the database

Usage

```
store_location(location, datafield, conn, hash, clean = TRUE)
```

Arguments

`location`	a data.frame with location metadata. Must contain the following columns: local_id, description, parent_local_id, datafield_local_id and external_code. Other columns are ignored.
`datafield`	a data.frame with datafield metadata. Must contain the variables local_id, datasource, table_name, primary_key and datafield_type. Other variables are ignored.
`conn`	a DBIconnection
`hash`	the hash of the update session
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

Details

- location must have variables local_id, description, parent_local_id, datafield_local_id and extranal_code. Other variables are ignored
- datafield must have variables local_id, datasource, table_name, primary_key and datafield_type
- all local_id variables must be unique within their data.frame
- all values in location$datafield_local_id must exist in datafield$local_id
- all values in location$parent_location must be either NA or exist in location$local_id

---

`store_location_group` *Store location groups*

Description

Store location groups

Usage

```
store_location_group(location_group, hash, conn, clean = TRUE)
```

Arguments

`location_group`	the data.frame with location groups. Must contains local_id, description and scheme. Other variables are ignored. local_id must have unique values.
`hash`	the hash of the update session
`conn`	a DBIconnection
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_location_group_location` *store the link between locations and location groups in the database*

Description

store the link between locations and location groups in the database

Usage

```
store_location_group_location(location_group_location, location_group,
  location, datafield, conn, hash, clean = TRUE)
```

Arguments

`location_group_location`	a data.frame with the locations per location group. Must contain location_group_local_id and location_local_id. Other columns are ignored.
`location_group`	the data.frame with location groups. Must contains local_id, description and scheme. Other variables are ignored. local_id must have unique values.
`location`	a data.frame with location metadata. Must contain the following columns: local_id, description, parent_local_id, datafield_local_id and external_code. Other columns are ignored.
`datafield`	a data.frame with datafield metadata. Must contain the variables local_id, datasource, table_name, primary_key and datafield_type. Other variables are ignored.
`conn`	a DBIconnection
`hash`	the hash of the update session
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

Details

- location_group_location must have variables location_local_id and location_group_local_id.
- location_group must have variables local_id, description and scheme
- location must have variables local_id, description, parent_local_id, datafield_local_id and extranal_code. Other variables are ignored
- datafield must have variables local_id, datasource, table_name, primary_key and datafield_type
- all local_id variables must be unique within their data.frame
- all values in location$datafield_local_id must exist in datafield$local_id
- all values in location$parent_location must be either NA or exist in location$local_id
- all values in location_group_location$location_local_id must exist in location$local_id
- all values in location_group_location$location_group_local_id must exist in location_group$local_id

`store_model_set` *Store model sets in the database*

Description

Store model sets in the database

Usage

```
store_model_set(model_set, hash, clean = TRUE, conn)
```

Arguments

`model_set`	a data.frame with the model sets. Must have variables "local_id", "description", "first_year", "last_year" and "duration". The variable "long_description" is optional.
`hash`	the hash of the update session
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.
`conn`	a DBIconnection

`store_model_type` *Store model type in the database*

Description

Store model type in the database

Usage

```
store_model_type(model_type, hash, clean = TRUE, conn)
```

Arguments

`model_type`	a data.frame with the modeltypes. Must have a variable "description". The variable "long_description" is optional.
`hash`	the hash of the update session
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.
`conn`	a DBIconnection

`store_n2kImport` *store an n2kImport object into the database*

Description

store an n2kImport object into the database

Usage

```
store_n2kImport(object, conn, hash, clean = TRUE)
```

Arguments

object	a n2kImport-class object
conn	a DBIconnection
hash	the hash of the update session
clean	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

store_n2kManifest — *store all models from an n2kManifest*

Description

store all models from an n2kManifest

Usage

```
store_n2kManifest(manifest, base, project, conn, status = "converged",
  hash, clean = TRUE)
```

Arguments

manifest	a n2kManifest-class
base	the base location to read the model
project	will be a relative path within the base location
conn	a DBIconnection
status	the status of the objects to be imported
hash	the hash of the update session
clean	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

store_n2kModel — *extract the results from an n2kModel and stored them*

Description

extract the results from an n2kModel and stored them

Usage

```
store_n2kModel(x, conn, hash, clean = TRUE)
```

Arguments

`x`	the n2kModel object
`conn`	a DBIconnection
`hash`	the hash of the update session
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_n2kResult` *store an n2kResult object into the database*

Description

store an n2kResult object into the database

Usage

```
store_n2kResult(object, conn, hash, clean = TRUE)
```

Arguments

`object`	a `n2kResult-class` object
`conn`	a DBIconnection
`hash`	the hash of the update session
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_observation` *store a datafield in the database*

Description

store a datafield in the database

Usage

```
store_observation(datafield, observation, location, parameter, conn, hash,
  clean = TRUE)
```

Arguments

`datafield`	a data.frame with datafield metadata. Must contain the variables local_id, datasource, table_name, primary_key and datafield_type. Other variables are ignored.
`observation`	a data.frame with observation metadata. Must contain the variables local_id, datafield_local_id, external_code, location_local_id, year and parameter_local_id. Other variables are ignored. datafield_local_id, external_code and parameter_local_id can be missing.
`location`	a data.frame with location metadata. Must contain the following columns: local_id, description, parent_local_id, datafield_local_id and external_code. Other columns are ignored.
`parameter`	a data.frame with parameters. Must contains the variables "description", "local_id", and "parent_parameter_local_id". Other variables are ignored.
`conn`	a DBIconnection
`hash`	the hash of the update session
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_parameter` *Store parameters*

Description

Store parameters

Usage

```
store_parameter(parameter, hash, conn, clean = TRUE)
```

Arguments

`parameter`	a data.frame with parameters. Must contains the variables "description", "local_id", and "parent_parameter_local_id". Other variables are ignored.
`hash`	the hash of the update session
`conn`	a DBIconnection
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_scheme` *Store a vector of schemes*

Description

Store a vector of schemes

Usage

```
store_scheme(scheme, hash, conn, clean = TRUE)
```

Arguments

`scheme`	the vector with scheme descriptions.
`hash`	the hash of the update session
`conn`	a DBIconnection
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_source_species` *store source species in the database*

Description

store source species in the database

Usage

```
store_source_species(source_species, datafield, conn, hash, clean = TRUE)
```

Arguments

`source_species`	a data.frame with source species metadata. Must contain local_id, description, datafield_local_id and extrenal_code. Other variables are ignored.
`datafield`	a data.frame with datafield metadata. Must contain variables local_id, datasource, table_name, primary_key and datafield_type.
`conn`	a DBIconnection
`hash`	the hash of the update session
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

store_source_species_species *store source species in the database*

Description

store source species in the database

Usage

```
store_source_species_species(species, language, source_species,
  source_species_species, datafield, conn, hash, clean = TRUE)
```

Arguments

`species`	a data.frame with species metadata. Must contain at least ‘local_id‘, ‘scientific_name‘ and ‘nbn_key‘. Other variable names must match the values in ‘language$code‘.
`language`	the data.frame with language Must contains code and description. Other variables are ignored. code and description must have unique values.
`source_species`	a data.frame with source species metadata. Must contain local_id, description, datafield_local_id and extrenal_code. Other variables are ignored.
`source_species_species`	as data.frame linking the local species id to the local source_species id. Must contain species_local_id and source_species_local_id. Other variables are ignored.
`datafield`	a data.frame with datafield metadata. Must contain variables local_id, datasource, table_name, primary_key and datafield_type.
`conn`	a DBIconnection
`hash`	the hash of the update session
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

store_species *store species in the database*

Description

store species in the database

Usage

```
store_species(species, language, conn, hash, clean = TRUE)
```

Arguments

`species`	a data.frame with species metadata. Must contain at least 'local_id', 'scientific_name' and 'nbn_key'. Other variable names must match the values in 'language$code'.
`language`	the data.frame with language Must contains code and description. Other variables are ignored. code and description must have unique values.
`conn`	a DBIconnection
`hash`	the hash of the update session
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_species_group` *Store species groups*

Description

Store species groups

Usage

```
store_species_group(species_group, hash, conn, clean = TRUE)
```

Arguments

`species_group`	the data.frame with species groups. Must contains local_id, description and scheme. Other variables are ignored. local_id must have unique values.
`hash`	the hash of the update session
`conn`	a DBIconnection
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_species_group_species` *store all species related information in the database*

Description

store all species related information in the database

Usage

```
store_species_group_species(species, language, source_species,
  source_species_species, datafield, species_group, species_group_species,
  hash, conn, clean = TRUE)
```

Arguments

`species`	a data.frame with species metadata. Must contain at least ‘local_id‘, ‘scientific_name‘ and ‘nbn_key‘. Other variable names must match the values in ‘language$code‘.
`language`	the data.frame with language Must contains code and description. Other variables are ignored. code and description must have unique values.
`source_species`	a data.frame with source species metadata. Must contain local_id, description, datafield_local_id and extrenal_code. Other variables are ignored.
`source_species_species`	as data.frame linking the local species id to the local source_species id. Must contain species_local_id and source_species_local_id. Other variables are ignored.
`datafield`	a data.frame with datafield metadata. Must contain variables local_id, datasource, table_name, primary_key and datafield_type.
`species_group`	the data.frame with species groups. Must contains local_id, description and scheme. Other variables are ignored. local_id must have unique values.
`species_group_species`	as data.frame linking the local species group id to the local species id. Must contain variables species_local_id and species_group_local_id. Other variables are ignored.
`hash`	the hash of the update session
`conn`	a DBIconnection
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.

`store_status` *Store status levels in the database*

Description

Store status levels in the database

Usage

```
store_status(status, hash, clean = TRUE, conn)
```

Arguments

`status`	a character vector with statuses
`hash`	the hash of the update session
`clean`	perform all database operations within a transaction and clean up the staging tables. Defaults to TRUE.
`conn`	a DBIconnection

`truncate_public` *Truncate all tables in the public schema: USE WITH CATION*

Description

Truncate all tables in the public schema: USE WITH CATION

Usage

```
truncate_public(conn)
```

Arguments

`conn`	a DBIconnection

Index