Package 'CommonDataModel'

October 1, 2024

Type Package

Title OMOP CDM DDL and Documentation Generator

Version 1.0.1

Maintainer Clair Blacketer <mblacke@its.jnj.com>

Description Generates the scripts required to create an Observational Medical Outcomes Partnership (OMOP) Common Data Model (CDM) database and associated documentation for supported database platforms. Leverages the 'SqlRender' package to convert the Data Definition Language (DDL) script written in parameterized Structured Query Language (SQL) to the other supported dialects.

License Apache License 2.0

Encoding UTF-8

Depends DatabaseConnector, SqlRender, rJava

Imports rmarkdown, stringr, DBI, dplyr, readr

Suggests knitr, testthat (>= 3.0.0), RSQLite, withr

NeedsCompilation no

RoxygenNote 7.2.3

Config/testthat/edition 3

Author Clair Blacketer [aut, cre]

Repository CRAN

Date/Publication 2024-10-01 20:20:02 UTC

Contents

`buildRelease` *Create OMOP CDM SQL files*

Description

Writes DDL, ForeignKey, PrimaryKey and index SQL files for given cdmVersion and targetDialect to the 'ddl' folder in specified output folder.

Usage

```
buildRelease(
  cdmVersions = listSupportedVersions(),
  targetDialects = listSupportedDialects(),
  outputfolder = file.path(tempdir(), "inst", "ddl")
)
```

Arguments

`cdmVersions`	The versions of the CDM you are creating, e.g. 5.3, 5.4. Defaults to all supported CDM versions.
`targetDialects`	A character vector of target dialects. Defaults to all supported dialects.
`outputfolder`	The base folder where the SQL files will be written. Subfolders will be created for each cdmVersion and targetDialect.

Value

Writes DDL, ForeignKey, PrimaryKey and index SQL files for given cdmVersion and targetDialect to the 'ddl' folder in specified output folder.

`buildReleaseZip` *Create OMOP CDM release zip*

Description

First calls `buildReleaseZips` for given cdmVersions and targetDialects. This writes the ddl sql files to the ddl folder. Then zips all written ddl files into a release zip to given output folder.

Usage

```
buildReleaseZip(
  cdmVersion,
  targetDialect = listSupportedDialects(),
  outputfolder = file.path(tempdir(), "output")
)
```

Arguments

`cdmVersion`	The version of the CDM you are creating, e.g. 5.3, 5.4. Defaults to all supported CDM versions.
`targetDialect`	The target dialect. Defaults to all supported dialects.
`outputfolder`	The output folder. Defaults to "output"

Details

If no (or multiple) targetDialect is given, then one zip is written with the files of all supported dialects.

Value

A character string containing the OHDSQL DDL

Examples

```
## Not run:
buildReleaseZip(cdmVersion='5.3', targetDialect='sql server', outputfolder='.')

## End(Not run)
```

`createDdl` *Create the OHDSI-SQL Common Data Model DDL code*

Description

The createDdl, createForeignKeys, and createPrimaryKeys functions each return a character string containing their respective DDL SQL code in OHDSQL dialect for a specific CDM version. The SQL they generate needs to be rendered and translated before it can be executed.

Usage

```
createDdl(cdmVersion)

createPrimaryKeys(cdmVersion)

createForeignKeys(cdmVersion)
```

Arguments

`cdmVersion`	The version of the CDM you are creating, e.g. 5.3, 5.4

Details

The DDL SQL code is created from a two csv files that detail the OMOP CDM Specifications. These files also form the basis of the CDM documentation and the Data Quality Dashboard.

Value

A character string containing the OHDSQL DDL

A string containing the OHDSQL for creation of primary keys in the OMOP CDM.

A string containing the OHDSQL for creation of foreign keys in the OMOP CDM.

Functions

- `createPrimaryKeys()`: createPrimaryKeys Returns a string containing the OHDSQL for creation of primary keys in the OMOP CDM.
- `createForeignKeys()`: createForeignKeys Returns a string containing the OHDSQL for creation of foreign keys in the OMOP CDM.

Examples

```
## Not run:
ddl <- createDdl("5.4")
pk <- createPrimaryKeys("5.4")
fk <- createForeignKeys("5.4")

## End(Not run)
```

`downloadCurrentDdl` *Get current DDL sitting on the main branch*

Description

Get current DDL sitting on the main branch

Usage

```
downloadCurrentDdl(
  githubPath = "OHDSI/CommonDataModel",
  pathToCsv = "Sql%20Server/OMOP%20CDM%20sql%20server%20ddl.txt",
  outputFile = paste0("inst/sql/sql_server/OMOP CDM ddl ", Sys.Date(), ".sql")
)
```

Arguments

`githubPath`	The path for the GitHub repo containing the package (e.g. 'OHDSI/CommonDataModel').
`pathToCsv`	The path for the snapshot inside the package.
`outputFile`	The path where the file should be saved.

Details

This function gets the current ddl on the CDM main branch. It will be taken from the Sql Server folder. The default location is `inst/settings/currentOmopDdl.sql`.

Value

The current DDL sitting on the main branch of the CommonDataModel repository.

Examples

```
## Not run:
downloadCurrentDdl("OHDSI/CommonDataModel",
 pathToCsv="Sql%20Server/OMOP%20CDM%20sql%20server%20ddl.txt")

## End(Not run)
```

`executeDdl` *Generate and execute the DDL on a database*

Description

This function will generate the DDL for a specific dbms and CDM version and then execute the DDL on a database.

Usage

```
executeDdl(
  connectionDetails,
  cdmVersion,
  cdmDatabaseSchema,
  executeDdl = TRUE,
  executePrimaryKey = TRUE,
  executeForeignKey = TRUE,
  ...
)
```

Arguments

`connectionDetails`	An object of class connectionDetails as created by the DatabaseConnector::createConnectionDetails function.
`cdmVersion`	The version of the CDM you are creating, e.g. 5.3, 5.4
`cdmDatabaseSchema`	The schema of the CDM instance where the DDL will be run. For example, this would be "ohdsi.dbo" when testing on sql server.
`executeDdl`	Should the DDL be executed? TRUE or FALSE
`executePrimaryKey`	Should the primary keys be added? TRUE or FALSE
`executeForeignKey`	Should the foreign keys be added? TRUE or FALSE
`...`	Other arguments passed on to DatabaseConnector::executeSql. (This allows the user to set the path to errorReportFile.)

Value

Writes the fully specified DDLs, primary keys, foreign keys, and indices to a file and then executes on a database.

Examples

```
## Not run:
executeDdl(connectionDetails = connectionDetails,
           cdmVersion = "5.4",
           cdmDatabaseSchema = "myCdm")

## End(Not run)
```

`listSupportedDialects` *List RDBMS dialects supported by this package*

Description

List RDBMS dialects supported by this package

Usage

```
listSupportedDialects()
```

Value

A list containing the supported Structured Query Language (SQL) dialects.

`listSupportedVersions` *List CDM versions supported by this package*

Description

List CDM versions supported by this package

Usage

```
listSupportedVersions()
```

Value

A character vector containing the supported Common Data Model (CDM) versions in major.minor format.

parseWiki *Parse Wiki files*

Description

Parses all .md files in the specified location (or any subfolders), extracting definitions of the Common Data Model.

Usage

```
parseWiki(mdFilesLocation, output_file)
```

Arguments

`mdFilesLocation`	Path to the root folder of the Wiki repository.
`output_file`	Path to where the output CSV file should be written.

Value

CSV files with the OMOP CDM specifications.

writeDdl *Write DDL script*

Description

Write the DDL to a SQL file. The SQL will be rendered (parameters replaced) and translated to the target SQL dialect. By default the @cdmDatabaseSchema parameter is kept in the SQL file and needs to be replaced before execution.

Usage

```
writeDdl(
  targetDialect,
  cdmVersion,
  outputfolder,
  cdmDatabaseSchema = "@cdmDatabaseSchema"
)

writePrimaryKeys(
  targetDialect,
  cdmVersion,
  outputfolder,
  cdmDatabaseSchema = "@cdmDatabaseSchema"
)

writeForeignKeys(
  targetDialect,
  cdmVersion,
  outputfolder,
  cdmDatabaseSchema = "@cdmDatabaseSchema"
)

writeIndex(
  targetDialect,
  cdmVersion,
  outputfolder,
  cdmDatabaseSchema = "@cdmDatabaseSchema"
)
```

Arguments

`targetDialect`	The dialect of the target database. Support dialects are specified by SqlRender::listSupportedDialects
`cdmVersion`	The version of the CDM you are creating, e.g. 5.3, 5.4
`outputfolder`	The directory or folder where the SQL file should be saved.
`cdmDatabaseSchema`	The schema of the CDM instance where the DDL will be run. For example, this would be "ohdsi.dbo" when testing on sql server. Defaults to "@cdmDatabaseSchema"

Value

Writes SQL file with the OMOP CDM DDL for the specified CDM version and target dialect in the output folder.

Writes a SQL file with the primary keys for the OMOP CDM based on the specified target dialect and CDM version.

Writes a SQL file with the foreign keys for the OMOP CDM based on the specified target dialect and CDM version.

Writes a SQL file with the indices for the OMOP CDM based on the specified target dialect and CDM version.

Functions

- `writePrimaryKeys()`: writePrimaryKeys Write the SQL code that creates the primary keys to a file.
- `writeForeignKeys()`: writeForeignKeys Write the SQL code that creates the foreign keys to a file.
- `writeIndex()`: writeIndex Write the rendered and translated sql that creates recommended indexes to a file.

Index