# Package ‘brendaDb’

November 28, 2024

**Type** Package

**Title** The BRENDA Enzyme Database

**Version** 1.20.0

**Description** R interface for importing and analyzing enzyme information from the BRENDA database.

**License** MIT + file LICENSE

**Encoding** UTF-8

**biocViews** ThirdPartyClient, Annotation, DataImport

**URL** `https://github.com/y1zhou/brendaDb`

**BugReports** `https://github.com/y1zhou/brendaDb/issues`

**Suggests** testthat, BiocStyle, knitr, rmarkdown, devtools

**Imports** dplyr, Rcpp, tibble, stringr, magrittr, purrr, BiocParallel, crayon, utils, tidyr, grDevices, rlang, BiocFileCache, rappdirs

**LinkingTo** Rcpp

**RoxygenNote** 7.2.3

**Roxygen** list(markdown = TRUE)

**SystemRequirements** C++11

**VignetteBuilder** knitr

**git_url** https://git.bioconductor.org/packages/brendaDb

**git_branch** RELEASE_3_20

**git_last_commit** e041c80

**git_last_commit_date** 2024-10-29

**Repository** Bioconductor 3.20

**Date/Publication** 2024-11-28

**Author** Yi Zhou [aut, cre] (<`https://orcid.org/0000-0003-0969-3993`>)

**Maintainer** Yi Zhou <`yi.zhou@uga.edu`>

# Contents

---

## `brendaDb-package` *brendaDb: the BRENDA enzyme database.*

---

### Description

brendaDb provides an R interface to download, clean and extract enzyme information from the BRENDA database.

### Details

The main aims of brendaDb include:

- Read text file downloaded from BRENDA into an R `tibble`
- Retrieve information for specific enzymes
- Query enzymes using their synonyms, gene symbols, etc.
- Query enzyme information for specific BioCyc pathways

To learn more about brendaDb, please refer to the vignette. `browseVignettes(package = "brendaDb")`

## Author(s)

**Maintainer**: Yi Zhou <`yi.zhou@uga.edu`> (ORCID)

## See Also

Useful links:

- `https://github.com/y1zhou/brendaDb`
- Report bugs at `https://github.com/y1zhou/brendaDb/issues`

---

# `acronyms` *Information fields and their corresponding acronyms.*

## Description

The RData file is generated by reading the entire BRENDA text file with `ReadBrenda()`, and getting unique values in the `field` column. The acronyms are extracted by the regex `^[A-Z_05]+`.

## Usage

```
data(acronyms)
```

## Format

A data.frame with 40 rows and 2 columns.

- field: field names in the BRENDA text file (all uppercase letters separated by underscores). This could be used in the `fields` argument of the `QueryBrenda()` function.
- acronym: acronyms used by the BRENDA file to indicate the fields.

---

# `CleanECNumber` *Remove deleted and transferred EC numbers.*

## Description

Some EC numbers have comments wrapped in parentheses. Most of them are deleted (in this case we remove them) entries or transferred (in this case we point to the new entry) entries.

## Usage

```
CleanECNumber(df)
```

## Arguments

| | |
|---|---|
| `df` | A `tibble` generated by `ReadBrenda()`. |

### Value

A `tibble` with deleted and transferred entries moved to the bottom, with columns:

- ID being the deleted/transferred ID,
- field being "TRANSFERRED_DELETED", and
- description being the information included in the original ID column.

### Examples

```
df <- ReadBrenda(system.file("extdata", "brenda_download_test.txt",
                             package = "brendaDb"))
brendaDb:::CleanECNumber(df)
```

---

## `ConfigBPCores` *Configure the number of cores used by BiocParallel.*

### Description

Configure the number of cores used by BiocParallel.

### Usage

```
ConfigBPCores(n.core = 0)
```

### Arguments

| | |
|---|---|
| `n.core` | Integer specifying the number of cores to use. Default is 0, which would result in using all available cores. |

### Value

The back-end of type `bpparamClass`.

### Examples

```
brendaDb:::ConfigBPCores(2)
```

---

`DownloadBrenda` *Download and unzip the BRENDA text file.*

---

## Description

By default, the function downloads a zipped BRENDA text file to a local cache directory, and extracts a `brenda_download.txt` file.

## Usage

```
DownloadBrenda(force.download = FALSE)
```

## Arguments

| | |
|---|---|
| `force.download` | Boolean value. If TRUE, ignore the cache and force re-download of the BRENDA text file. Default is FALSE. |

## Value

A string of the path to the downloaded BRENDA text file.

## Examples

```
## Not run: DownloadBrenda()
```

---

`ExtractField` *Extract a specific field from a brenda.entries object.*

---

## Description

Retrieve one field from all the brenda.entry objects. A column of EC numbers will be added to distinguish different enzymes.

## Usage

```
ExtractField(res, field, entries = NULL)
```

## Arguments

| | |
|---|---|
| `res` | A brenda.entries object from `QueryBrenda()`. |
| `field` | A string indicating the field to extract. Nested fields should be separated by `$`, e.g. `parameters$ph.optimum`. |
| `entries` | A character vector with values of EC numbers. This should be a subset of `names(res)`. |

## Value

A tibble with all columns from the tibble in the given field, and extra columns containing the EC numbers and organisms.

## Examples

```
df <- ReadBrenda(system.file("extdata", "brenda_download_test.txt",
                             package = "brendaDb"))
res <- QueryBrenda(brenda = df, EC = c("1.1.1.1", "6.3.5.8"),
                   n.core = 2)
ExtractField(res, field = "molecular$stability$general.stability")
ExtractField(res, field = "structure$subunits")
```

---

# `ID2Enzyme` *A helper function for converting names/synonyms to EC numbers.*

## Description

A helper function for converting names/synonyms to EC numbers.

## Usage

```
ID2Enzyme(brenda, ids)
```

## Arguments

| | |
|---|---|
| `brenda` | A `tibble` generated from `ReadBrenda()`. |
| `ids` | A character vector of IDs to be converted. |

## Details

The function goes through "RECOMMENDED_NAME", "SYSTEMATIC_NAME", and "SYNONYMS" in the BRENDA file, and uses regexes to look for the given IDs. Values in the three columns are kept if the regex had a hit, otherwise NA is filled. The function can take in IDs of multiple sources, e.g. `c("ADH4", "CD38", "pyruvate dehydrogenase")`. Note that using aliases instead of symbols could lead to false positives in the output table.

## Value

A tibble with columns ID, EC, and at least one of (RECOMMENDED_NAME, SYSTEMATIC_NAME and SYNONYMS).

## Examples

```
df <- ReadBrenda(system.file("extdata", "brenda_download_test.txt",
                             package = "brendaDb"))
ID2Enzyme(df, c("CD38", "ADH4", "pyruvate dehydrogenase"))
```

---

## `InitBrendaDeprecatedEntry` *Create a* `brenda.deprecated.entry` *object.*

---

### Description

Some EC numbers are transferred or deleted. For these entries, return a `brenda.deprecated.entry` object and the corresponding message.

### Usage

```
InitBrendaDeprecatedEntry(EC, msg)

is.brenda.deprecated.entry(x)
```

### Arguments

| | |
|---|---|
| `EC` | A string indicating EC number of the enzyme. |
| `msg` | A string of the transferred / deleted message. |
| `x` | Any object. |

### Details

`is.brenda.deprecated.entry` checks if an object is a `brenda.deprecated.entry` object. If the input is a `brenda.entries` object, check all items in the list to see if they are `brenda.deprecated.entry` objects and return a boolean vector of the same length.

### Value

A `brenda.deprecated.entry` obeject.

A boolean vector of the same length as `x`.

### Examples

```
brendaDb:::InitBrendaDeprecatedEntry("6.3.5.8", "Transferred to EC 2.6.1.85")
df <- ReadBrenda(system.file("extdata", "brenda_download_test.txt",
                             package = "brendaDb"))
is.brenda.entry(QueryBrenda(df, "6.3.5.8"))
```

---

`InitBrendaEntry` *Create a* `brenda.entry` *object.*

---

## Description

The list should contain 6 sublists: `nomenclature`, `interactions`, `parameters`, `molecular`, `stability`, and `bibliography`. All sublists should be empty by default apart from `EC` under `nomenclature`.

## Usage

```
InitBrendaEntry(
  EC,
  protein = NA,
  systematic.name = NA,
  recommended.name = NA,
  synonyms = NA,
  reaction = NA,
  reaction.type = NA,
  substrate.product = NA,
  natural.substrate.product = NA,
  cofactor = NA,
  metals.ions = NA,
  inhibitors = NA,
  activating.compound = NA,
  km.value = NA,
  turnover.number = NA,
  ki.value = NA,
  pi.value = NA,
  ph.optimum = NA,
  ph.range = NA,
  temperature.optimum = NA,
  temperature.range = NA,
  specific.activity = NA,
  ic50 = NA,
  source.tissue = NA,
  localization = NA,
  general.stability = NA,
  storage.stability = NA,
  ph.stability = NA,
  organic.solvent.stability = NA,
  oxidation.stability = NA,
  temperature.stability = NA,
  purification = NA,
  cloned = NA,
  engineering = NA,
  renatured = NA,
  application = NA,
  molecular.weight = NA,
  subunits = NA,
  posttranslational.modification = NA,
  crystallization = NA,
```

```
  bibliography = NA
)

is.brenda.entry(x, verbose = FALSE)
```

### Arguments

| | |
|---|---|
| `EC` | A string indicating EC number of the enzyme. |
| `protein` | The description string of a PR field. |
| `systematic.name` | The description string of a SN field. |
| `recommended.name` | The description string of a RN field. |
| `synonyms` | The description string of a SY field. |
| `reaction` | The description string of a RE field. |
| `reaction.type` | The description string of a RT field. |
| `substrate.product` | The description string of a SP field. |
| `natural.substrate.product` | The description string of a NSP field. |
| `cofactor` | The description string of a CF field. |
| `metals.ions` | The description string of a ME field. |
| `inhibitors` | The description string of a IN field. |
| `activating.compound` | The description string of a AC field. |
| `km.value` | The description string of a KM field. |
| `turnover.number` | The description string of a TN field. |
| `ki.value` | The description string of a KI field. |
| `pi.value` | The description string of a PI field. |
| `ph.optimum` | The description string of a PHO field. |
| `ph.range` | The description string of a PHR field. |
| `temperature.optimum` | The description string of a TO field. |
| `temperature.range` | The description string of a TR field. |
| `specific.activity` | The description string of a SA field. |
| `ic50` | The description string of a IC50 field. |
| `source.tissue` | The description string of a ST field. |
| `localization` | The description string of a LO field. |
| `general.stability` | The description string of a GS field. |
| `storage.stability` | The description string of a SS field. |
| `ph.stability` | The description string of a PHS field. |

| | |
|---|---|
| `organic.solvent.stability` | The description string of a OSS field. |
| `oxidation.stability` | The description string of a OS field. |
| `temperature.stability` | The description string of a TS field. |
| `purification` | The description string of a PU field. |
| `cloned` | The description string of a CL field. |
| `engineering` | The description string of a EN field. |
| `renatured` | The description string of a REN field. |
| `application` | The description string of a AP field. |
| `molecular.weight` | The description string of a MW field. |
| `subunits` | The description string of a SU field. |
| `posttranslational.modification` | The description string of a PM field. |
| `crystallization` | The description string of a CR field. |
| `bibliography` | The description string of a RF field. |
| `x` | Any object. |
| `verbose` | Boolean value default to FALSE. If TRUE, prints message when all elements are brenda.entries. |

## Details

`is.brenda.entry` checks if an object is a `brenda.entry` object. If the input is a `brenda.entries` object, check all items in the list to see if they are `brenda.entry` objects and return a boolean vector of the same length.

## Value

A `brenda.entry` object with all fields other than nomenclature$ec being NA.

A boolean vector of the same length as `x`.

## Examples

```
brendaDb:::InitBrendaEntry("1.1.1.100")
df <- ReadBrenda(system.file("extdata", "brenda_download_test.txt",
                             package = "brendaDb"))
is.brenda.entry(QueryBrenda(df, "6.3.5.8"))
```

---

## `ParseGeneric` *Generic parser for a description string.*

### Description

Descriptions are generally structured as the following:

- Protein information is included in '#'...#',
- Literature citations are in '<...>',
- Commentaries in '(...)', and
- field-special information in '...'.

This function separates these fields into different columns.

### Usage

```
ParseGeneric(description, acronym)
```

### Arguments

| | |
|---|---|
| `description` | A description string from one of the entries. |
| `acronym` | The acronym of the field. Can be found with `ShowFields()`. |

### Details

The `description` column contains values extracted by BRENDA in each field.

The `fieldInfo` column contains different information in different fields:

- In a `SYNONYMS` entry, it is either the source of the identifier, or part of the description (a false positive).
- In `KM_VALUE`, `TURNOVER_NUMBER` entries, it's the corresponding substrate.

### Value

A `tibble` with columns: proteinID, description, fieldInfo, commentary, and refID

---

## `ParseNoDescription` *Generic parser for a description string without extracted values.*

### Description

This parser works for fields `storage.stability`, `general.stability`, `oxidation.stability`, `cloned`, `purification`, `crystallization` and `renatured`.

These fields in BRENDA don't have extracted values - the commentary itself is the extracted value.

### Usage

```
ParseNoDescription(description, acronym)
```

### Arguments

| | |
|---|---|
| `description` | A description string from one of the entries. |
| `acronym` | The acronym of the field. Can be found with `ShowFields()`. |

### Value

A `tibble` with columns: proteinID, description and refID.

---

## `ParseProtein` *Parse a "PROTEIN" entry.*

### Description

Expand the string into a `tibble`.

### Usage

```
ParseProtein(description)
```

### Arguments

| | |
|---|---|
| `description` | The description string in a "PROTEIN" entry. |

### Value

A `tibble` with five columns: proteinID, description, uniprot, commentary and reference. The description column is the source organism.

### Examples

```
x <- paste0(
"PR\t#1# Cavia porcellus  (#1# SULT1A2 <1,2,6,7>) <1,2,6,7>\n",
"PR\t#2# Mus musculus <11,18,19>\n")
brendaDb::ParseProtein(x)
```

---

## `ParseProteinNum` *Parse protein information strings or reference strings.*

### Description

Given a string like "#1,45,72#", parse into a character vector of c("1", "45", "72"). Consecutive commas are collapsed into one, and spaces are treated as commas.

### Usage

```
ParseProteinNum(x, type)
```

### Arguments

| | |
|---|---|
| `x` | A string in the format of "#1#" or "#1,2,3#" or "<1,3>". |
| `type` | Either "protein" or "reference". |

### Value

A string, or a vector of strings of protein numbers.

### Examples

```
brendaDb:::ParseProteinNum("#1,2,3#", "protein")
# [1] "1,2,3"
brendaDb:::ParseProteinNum("<123>", "reference")
# [1] "123"
```

---

## `ParseReaction` *Parser for the description strings of certain reaction-related fields.*

### Description

This parser works for the fields `substrate.product` and `natural.substrate.product`.

### Usage

```
ParseReaction(description, acronym)
```

### Arguments

| | |
|---|---|
| `description` | A description string from one of the entries. |
| `acronym` | The acronym of the field. Can be found with `ShowFields()`. |

### Details

The reversibility of the reactions are wrapped in ; substrates and products are separated by "="; commentaries on substrates are wrapped in "()", and commentaries on products are wrapped in "||".

### Value

A `tibble` with columns: proteinID, substrate, product, commentarySubstrate, commentaryProduct, fieldInfo and refID

---

## `ParseRecommendedName` *Parse a "RECOMMENDED_NAME" entry into a string.*

---

### Description

Remove useless characters in the description.

### Usage

```
ParseRecommendedName(description)
```

### Arguments

| | |
|---|---|
| `description` | The description string in a "RECOMMENDED_NAME" entry. |

### Value

A string to fill into the `recommended.name` field in `brenda.nomenclature`.

### Examples

```
x <- "RN\tD-arabinose 1-dehydrogenase (NAD+)"
brendaDb:::ParseRecommendedName(x)
```

---

## `ParseReference` *Parse a "REFERENCE" entry.*

---

### Description

Expand the string into a `tibble`.

### Usage

```
ParseReference(description)
```

### Arguments

| | |
|---|---|
| `description` | The description string in a "REFERENCE" entry. |

### Value

A `tibble` with three columns: refID, title and pubmed.

## Examples

```
x <- paste0(
"RF\t<1> Talbot, B.G.; Thirion, J.P.: Purification\n\t",
"and properties of two distinct groups of ADH isozymes from Chinese\n\t",
"hamster liver. Biochem. Genet. (1981) 19, 813-829. {Pubmed:6794566}\n",
"RF\t<12> Woronick, C.L.: Alcohol dehydrogenase from human liver. Methods\n\t",
"Enzymol. (1975) 41B, 369-374. {Pubmed:236461} (c,review)\n",
"RF\t<10> Herrera, E.; Zorzano, A... {Pubmed:} (c,review)\n")
brendaDb:::ParseReference(x)
```

---

# `ParseSystematicName` *Parse a "SYSTEMATIC_NAME" entry into a string.*

---

## Description

Remove useless characters in the description.

## Usage

```
ParseSystematicName(description)
```

## Arguments

| | |
|---|---|
| `description` | The description string in a "SYSTEMATIC_NAME" entry. |

## Value

A string to fill into the `systematic.name` field in `brenda.nomenclature`.

## Examples

```
x <- "SN\talcohol:NAD+ oxidoreductase"
brendaDb:::ParseSystematicName(x)
```

---

# `PrettyPrintBrendaEntry` *Print a brenda.entry in a tree view.*

---

## Description

Print a brenda.entry in a tree view.

## Usage

```
PrettyPrintBrendaEntry(x, index, tail.idx, depth, full.output)
```

### Arguments

| | |
|---|---|
| `x` | A `brenda.entry` object. |
| `index` | A string of the name of the sublist. |
| `tail.idx` | A string showing the last element in the entry. This is for printing a different character in the tree. |
| `depth` | Int, showing the depth of the element. |
| `full.output` | A boolean default to FALSE. If TRUE, print entry even if it's empty (NA or 0 rows). |

### Value

Nothing; prints object tree-view to the terminal.

---

## `print.brenda.entries` *Show the number of regular and transferred/deleted brenda.entry objects in the brenda.entries list.*

### Description

Show the number of regular and transferred/deleted brenda.entry objects in the brenda.entries list.

### Usage

```
## S3 method for class 'brenda.entries'
print(x, ..., verbose = FALSE)
```

### Arguments

| | |
|---|---|
| `x` | A brenda.entries list returned by `QueryBrenda()`. |
| `...` | Other arguments passed to the generic function. |
| `verbose` | Boolean; if TRUE, print tree views of brenda.query objects. |

### Value

Nothing; print summary information to the terminal.

---

## `print.brenda.entry` *Show the non-empty fields in the query result.*

### Description

For details, see `PrettyPrintBrendaEntry()`.

### Usage

```
## S3 method for class 'brenda.entry'
print(x, full.output = FALSE, ...)
```

### Arguments

| | |
|---|---|
| `x` | A brenda.entry object (elements in the list returned by the function `QueryBrenda()`). |
| `full.output` | A boolean default to FALSE. If TRUE, include all entries even if they are empty (NA or 0 rows). |
| `...` | Other arguments passed to the generic function. |

### Value

Nothing; print object information to the terminal.

---

## `PrintTreeHelper` *Print the tree structure with correct whitespace.*

### Description

Print the tree structure with correct whitespace.

### Usage

```
PrintTreeHelper(index, tail.idx, depth)
```

### Arguments

| | |
|---|---|
| `index` | A string of the name of the sublist. |
| `tail.idx` | A string showing the last element in the entry. This is for printing a different character in the tree. |
| `depth` | Int, showing the depth of the element. |

### Value

Nothing; prints tree structure to the terminal.

---

`QueryBrenda` *Query for multiple enzymes.*

---

### Description

Use a vector of EC numbers to retrieve information from the BRENDA `tibble` read in by `ReadBrenda()`. Invalid EC numbers will be removed and a message will be generated.

### Usage

```
QueryBrenda(brenda, EC, n.core = 0, fields = FALSE, ...)
```

### Arguments

| | |
|---|---|
| `brenda` | A `tibble` containing information from BRENDA. |
| `EC` | A string of the EC number. |
| `n.core` | Integer specifying the number of cores to use. Default is 0, which would result in using all available cores. |
| `fields` | A character vector indicating fields to parse. Default is FALSE, which would be returning all fields. |
| `...` | Other parameters passed to `QueryBrendaBase()`. |

### Value

A list of `brenda.entry` objects.

### See Also

`QueryBrendaBase()` `ConfigBPCores()` `SelectOrganism()`

### Examples

```
df <- ReadBrenda(system.file("extdata", "brenda_download_test.txt",
                          package = "brendaDb"))
res <- QueryBrenda(brenda = df, EC = c("1.1.1.1", "1.1.1.10", "6.3.5.8"),
                   n.core = 2, organisms = "Homo sapiens")
```

---

`QueryBrendaBase` *Query for a specific enzyme.*

---

### Description

Use a EC number to retrieve information from the BRENDA `tibble` read in by `ReadBrenda()`.

### Usage

```
QueryBrendaBase(brenda, EC, organisms = FALSE)
```

## Arguments

| | |
|---|---|
| `brenda` | A `tibble` containing information from BRENDA. |
| `EC` | A string of the EC number. |
| `organisms` | A character vector indicating organisms to keep. Default is FALSE, which would keep all organisms. |

## Value

A `brenda.entry` object.

## See Also

`ReadBrenda()` `InitBrendaEntry()`

## Examples

```
df <- ReadBrenda(system.file("extdata", "brenda_download_test.txt",
                             package = "brendaDb"))
brendaDb:::QueryBrendaBase(brenda = df, EC = "1.1.1.1",
                              organisms = "Homo sapiens")
```

---

# `ReadBrenda` *Read BRENDA text file into matrix.*

## Description

For each EC entry, split the annotations into three columns:

- ID: EC number, e.g. 1.1.1.1
- field: the content of the information, e.g. protein, localization
- description: everything else

## Usage

```
ReadBrenda(filepath, clean = TRUE)
```

## Arguments

| | |
|---|---|
| `filepath` | A string indicating the path to the text file. |
| `clean` | Boolean; if TRUE, run `CleanECNumber()` after reading the file. |

## Value

A matrix containing information about the EC entries.

## Examples

```
brenda_txt <- system.file("extdata", "brenda_download_test.txt",
                          package = "brendaDb")
df <- ReadBrenda(brenda_txt)
```

---

## `ReadBrendaFile` *Read raw BRENDA text file.*

---

### Description

Read file into buffer, and load all non-empty lines. Comment lines (starting with *) are skipped. The text file should be downloaded from https://www.brenda-enzymes.org/download_brenda_without_registration.php

### Usage

```
ReadBrendaFile(filepath)
```

### Arguments

| | |
|---|---|
| `filepath` | A string indicating the path to the text file. |

### Value

A vector with each element being a line in the file.

---

## `SelectOrganism` *Select a subset of organisms in the brenda.query object.*

---

### Description

Select a subset of organisms in the brenda.query object.

### Usage

```
SelectOrganism(query, org.id)
```

### Arguments

| | |
|---|---|
| `query` | A brenda.entry object from `QueryBrendaBase()`. |
| `org.id` | A named character vector with values as `proteinID`s. |

### Value

A subset of the given brenda.query object with the given organisms selected.

### See Also

`QueryBrendaBase()`

---

## `SeparateEntries` *Convert vector of lines to matrix.*

### Description

For each EC entry, split the annotations into three columns:

- ID: EC number, e.g. 1.1.1.1
- field: the content of the information, e.g. protein, localization
- description: everything else

### Usage

```
SeparateEntries(lines)
```

### Arguments

| | |
|---|---|
| `lines` | The output vector from `read_brenda_file`. |

### Value

A vector<vector> containing information about the EC entries. In R this is a list of 3 lists.

---

## `SeparateSubentries` *Preprocessing of entry description.*

### Description

Separate subentries and strip unnecessary whitespace.

### Usage

```
SeparateSubentries(description, acronym = NA)
```

### Arguments

| | |
|---|---|
| `description` | The `description` string of the field. |
| `acronym` | The acronym of the field. Can be found with `ShowFields()`. |

### Value

A list of strings with each subentry as an element.

### Examples

```
x <- "SN\talcohol:NAD+ oxidoreductase"
brendaDb:::SeparateSubentries(x, "SN")
```

---

`ShowFields` *Show all unique BRENDA fields and their corresponding acronyms.*

---

## Description

Show all unique BRENDA fields and their corresponding acronyms.

## Usage

```
ShowFields(df)
```

## Arguments

| | |
|---|---|
| `df` | A data.frame with columns "field" and "description" |

## Value

A data.frame with columns "field" and "acronym".

## Examples

```
df <- ReadBrenda(system.file("extdata", "brenda_download_test.txt",
                             package = "brendaDb"))
ShowFields(df)
```

# Index