Package: gmDatabase (via r-universe)

October 24, 2024

Version 0.5.0

Date 2016-05-31

Title Accessing a Geometallurgical Database with R

Author K. Gerald van den Boogaart [aut, cre], Stephan Matos Camacho [aut]

Maintainer K. Gerald van den Boogaart <support@boogaart.de>

Depends R(>= 2.1.0), DBI, RMySQL, foreach, methods, digest, shiny

Suggests MASS

Description A template for a geometallurgical database and a fast and easy interface for accessing it is provided in this package.

License GPL (>= 2) | LGPL (>= 2)

LazyLoad TRUE

URL http://www.r-project.org, http://www.stat.boogaart.de

Copyright (C) 2016 by Helmholtz Institute Freiberg for Resource Technology

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-06-16 12:32:05

Contents

---

dbNow *Get the actual system time*

Description

dbNow retrieves the acrtual system time.

Usage

```
dbNow()
```

Details

Returns the actual system time in the format yyyy-mm-dd hh:mm:ss.

Value

A string.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

Examples

```
dbNow()
```

`extractDollarExpressions` *Extracting a dollar expression*

Description

Extracts the expression containing a dollar symbol.

Usage

```
extractDollarExpressions(EXPR)
```

Arguments

`EXPR`	an R expression

Details

`extractDollarExpressions` extracts the expression containing a dollar symbol. Furthermore it provides an alias for it. This function is meant for internal use in `gmReadInternal`, when dollar expressions occur in bracket terms.

Value

Returns a structure, containing the whole expression using the alias instead of the original dollar expression in `expr` and the dollar expression itself in `dollarExpr`.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

See Also

`gmReadInternal`

Examples

```
extractDollarExpressions(quote(root$project == 2))
```

findVarsInExpression findVarsInExpression

Description

parses expressions and returns the variables

Usage

```
findVarsInExpression(EXPR)
```

Arguments

EXPR	a call object representing an R expression

Details

findVarsInExpression takes a call and returns the names. It is used in gmReadInternal for parsing an R expression of an SQL query for variables.

Value

A string consisting of the variables.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

See Also

replaceVarsInExpression, gmReadInternal

Examples

```
a <- quote( A==B^C+A^2+C )
findVarsInExpression(a)
```

forKeyVal *forKeyVal – Looping named lists*

Description

forKeyVal loops a named list or vector, with one variable bound to the key and another to the value.

Usage

```
forKeyVal(key,val,LIST,block,envir=parent.frame())
```

Arguments

key	the variable to be bound by the name of the element of the LIST
val	the variable to be bound by the value of the element of the LIST
LIST	a list
block	the block to be exectued, with this variables bound
envir	the environment in which the variables are bound and the block is executed

Details

It works much like a `for(val in LIST) block` with the difference. It however additionally binds key to the value name of the list element, but does not recognize continue or break statements. This might change at some point in future.
Empty or unbound names result in a binding of key to `NULL`.

Value

the value of the last execution of the block

Author(s)

K. Gerald van den Boogaart

See Also

`for`, `lapply`

Examples

```
forKeyVal(name,x,c(a=1,b=5,c=6),{
  cat(name,"=>",x,"\n")
})

forKeyVal(name,x,list(a=4,b=1:7,c=c(a="Aber",b="nicht")), {
  cat(name,"\n",sep="")
  if(is.null(names(x)))
```

```
    names(x) <- 1:length(x)
  forKeyVal(iname,x,x,{
    cat(name,".",iname,"=>",x,"\n",sep="")
  })
})
```

---

gmAdd *Adding and updating objects in a geometallurgical database.*

Description

gmAdd adds a new object to a geometallurgical database. Its attributes can be set and updated with gmSet

Usage

```
gmAdd(where, what, data, rg=getOption("defaultRightsGroup"), EXPR=substitute(where),
  envir=parent.frame(), db=getOption("gmDB"), force=FALSE)
gmSet(where, data, varID=NULL, childs=NULL, members=NULL,EXPR=substitute(where),
  envir=parent.frame(), db=getOption("gmDB"), update=TRUE)
```

Arguments

where	an expression specifying where in the hierarchical structure of the database the new object(s) should be added.
what	specifies the variable type of the new object(s). Apart from an R expression of an SQL query or a gmExpr, a ID from the database is possible, too.
data	list or data frame giving attributes for the new object(s).
rg	states the id of the rights groups this object will belongs to.
varID	the variable ID of the objects to be updated
childs	a data frame containing the class member of the object to be updated
members	a vector containing the names of the class members of what, which will be updated
EXPR	the quoted version of where.
envir	defines the environment used for evaluation of where.
db	defines the database, on which the query will be run.
force	logical: if FALSE, the case of missing required class members results in an abortion, otherwise only warnings are printed.
update	logical: if TRUE, set objects will be updated instead of adding new entries.

Details

gmAdd adds one or more new objects of the specified variable name to a geometallurgical database. The number of new objects is determined by data. This is a list or data frame stating values for the class members of each new object. If force=TRUE missing required class members are ignored and a warning is given. If force=FALSE the insert is aborded in the case of missing required members with a corresponding error message. If no rights group is given the object will not be linked to the default rights group. If that is not explicitly set by gmDefaultRightsGroup the new object will not be linked to any rights group. Therefore it will be accessible by everyone.

For the actual insertion of the values of the members, the function gmSet is used. If a member given in data is already set for this object, then the database is being updated. Otherwise this attribute is added. Attributes which are not allowed for this object result in an abortion and an error message.

If gmSet is used outside gmAdd the argmuments varID, childs and members may be missing and therefore set to NULL. Then they are queried prior to the insertion. This setting is used to determine, if gmSet is called by gmAdd. If data contains a set object, setting the argument update to TRUE prohibits updating this member. Instead an additional entry is added.

If gmSet is called by gmAdd and a creationTime is missing in data, gmSet adds the actual time instead. Since that for variables of the type set more than one entry is permitted, you have the possibility of updating the entry or adding a new reference by setting update=FALSE. If there is already more than one entry, updating is not permitted.

Value

gmAdd returns the gmID of the first added object.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

See Also

gmRead, gmExpr

Examples

```
## Not run:
  gmAdd(root,"project",list(gmTitle="sunny thursday", gmCreator=3))
  gmSet(root$project[gmTitle=="sunny thursday"], list(gmName="something"))

## End(Not run)
```

---

gmAndify *Creating a conjunction*

Description

gmAndify creates a call, which is a conjunction of list elements given in its arguments for further use in ON or WHERE clauses of SQL statements

Usage

```
gmAndify(l,and="&")
```

Arguments

l	list of elements, which shall be conjuncted
and	determines the kind of conjunction being used. Default case is the logical AND.

Details

Instead of a conjunction you can create a disjunction using and="|". This function is in gmReadInternal

Value

A string of class call for further use.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

See Also

gmRead

Examples

```
gmAndify(c("a","b"))
```

---

gmChangePassword *Function for changing the password of a user in the geometallurgical database.*

Description

A gui for changing the password of user in the database.

Usage

```
gmChangePassword(db=getOption("gmDB"))
```

Arguments

db	A database connection object as returned by gmConnectServer.

Details

gmChangePassword allows the user to change the password for accessing data in the database. It uses shiny to provide aa gui, that hides the letters of an entered password.

Author(s)

S. Matos Camacho

See Also

shiny

Examples

```
## Not run:
gmChangePassword()

## End(Not run)
```

---

gmClass *Returning information from geometallurgicala database*

Description

Get information on given object or variable from the database.

Usage

```
gmClass(expr, var, envir=parent.frame(), EXPR=substitute(expr),db=getOption("gmDB"))
gmClassMembers(expr, var, envir=parent.frame(), EXPR=substitute(expr),
               db=getOption("gmDB"))
```

Arguments

expr	R expression of an SQL query. Expressions created with gmExpr can be used as well.
var	a number or variable name specifying a variable.
envir	defines the environment used for evaluation of expr
EXPR	The quoted version of such an expression.
db	defines the database, on which the query will be run.

Details

If you want information on a special variable in the database, specify var. You may use its variable id or the variable name. If var is missing information on the object(s) specified by expr is retrieved.

gmClass gives information on the variable itself or the variable the object is. gmClassMembers shows information about all possible members of a variable or the variable the object is.

Value

gmClass returns a data frame containing the id, gmVarID, gmVarTypeID, gmVarName, and gmVarDescription of the desired variable or object. If the object is not distinct, then information for every object retrieved. gmClassMembers returns a data frame containing the gmVarID and gmVarName of this all of the possible given members of variable respectively object. required is a boolean stating if the member is mandatory or not, definer states the variable, where this member is defined. If it is not inheritated, then the variable itself is given.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

See Also

gmGet

Examples

```
## Not run:
.gmDB <- dbConnect(...)
root <- "root"
gmClass(root)
## returns information on the root in the database

gmClass(root$project)
## returns information on every project in root in the database

gmClass(var="project")
## returns information on the variable/class project

gmClass(var=2)
## returns information on the variable with variable id 2

## End(Not run)
```

---

gmConnectServer *Geometallurgy Database connection*

Description

A geometallurgy database is a database on a relational database server. It can handle very general and recursive information for the transparent access to whole projects from a statistical analysis software.

Usage

```
gmConnectServer(..., server="localhost", dbuser, asDefault=TRUE)
gmDisconnectServer(db=getOption("gmDB"))
gmCreateDB(..., dbName="gmDatabase", admin=NULL, adminPwd=NULL)
gmListVariables(pattern="%",db=getOption("gmDB"))
gmListVariableTypes(db=getOption("gmDB"))
gmDBSummary(db=getOption("gmDB"))
gmEscapeStrings(s,db=getOption("gmDB"))
gmRequest(fun, parameters, db)
```

Arguments

...	Arguments to dbConnect from RMySQL to contact the database server.
asDefault	Logical indicating whether being used as default connection.
db	A database connection object as returned by gmConnectServer.
server	The address of the server, where the R server.
dbuser	A list giving the login and password of the user working in the geometallurgical database.
dbName	String stating the name for the database to-be.
admin	The login for the administrator/the first database user.
adminPwd	The password for the administrator/the database user.
pattern	An SQL-regular expression (for LIKE) to specify the variable names to be listed.
s	A character vector of strings to be escaped.
fun	A function name.
parameters	A list of parameters for the function fun

Details

The geometallurgy database provides an abstraction layer to a relational database storage of many eventually differently structured datasets, which might contain datasets recursively. Only in this very complex structure it is possible to respresent the data needed for typical statistical tasks of geometallurgy.

The data storage starts from a common root object "root", which contains a variable set variable "project".

The Database connection is handled by gmConnectServer and gmDisconnectServer. In order to assure a proper rights management, the user does not directly connect to the database. We implemented a two level security system. The first level represents the connection to the database server. Therefore a general user on the server is needed, who is provided with the required rights to access and manipulate the database itself. The corresponding credentials are compiled in ... and will be forwarded to dbConnect from the package RMySQL. Data access happens on the second level, where the dbuser is involved. His credentials are checked against the information stored in the database. Everytime some information is queried from the database, the result is given according to the rights this user owns on this particular data. If no user is explicitly given during the function call, i.e. dbuser is missing, gmConnectServer will start a dialog to access the (db)user's credentials. Same is true, if information on the general database server user is omitted.

In order to create a compatible database, you may gmCreateDB. A connection to the given MySQL-server is established and a new database of the given dbName is created. Furthermore first variables to realise the user/rights management are provided. Additonally a first user/admin is added to the database with the given username and password, so that this user is able to login to the database afterwards. If no MySQL login, MySQL password or MySQL host is given, a shiny app requests these information. Same happens if no login or password is given for the first user/admin.

This package can be used in a client - server approach. Then gmConnectServer will not return a database connection, but a socket connection. It will be used to send the commands to an R server. The actual translation from R into SQL is then done on the server. Again only the accessible data according to the user's rights will be returned. In order to activate the client-server mode, use the argument server to state the server you want to access. By default server is set to localhost.

gmRequest is an internal function. It should not be called directly. Depending on the class of database connection used, it decides if the fun is evaluated locally or handed over to the server.

Value

gmConnectServer
: Returns a server connection to be used in subsequent calls. It is assigned to the hidden variable .gmDB. If server ist set to localhost, it will be a database connection instead.

gmDisconnectServer
: Returns the same as gmDisconnectDB. Additionally .gmDB is set to zero.

gmCreateDB
: No return value.

gmListVariables
: Returns a dataframe describing the variables defined in the geometallurgy database.

gmListVariableTypes
: Returns a dataframe describing the variable types defined in the geometallurgy database.

gmDBSummary
: Returns a dataframe giving variable name, variable type and number of objects for each variable.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

See Also

MySQLConnection-class

Examples

```
## Not run:
gmCreateDB(MySQL(), user="mysqlUser", password="mysqlPassword", host="mysqlServer",
  dbName="myDB", admin="admin",adminPwd="myAdminPassword")
gmConnectServer(MySQL(), group="test", server="localhost")
gmListVariables(pattern=".*",db=.gmDB)
gmListVariableTypes(db=.gmDB)
gmDBSummary(db=.gmDB)
gmDisconnectServer()
```

```
## End(Not run)
```

---

gmCreateClass *Providing new classes and adding new users, rights groups, usergroups, etc.*

Description

For working with a (geometallurgical) database you need at least some users, rights groups to manage their access rights, or classes/variables, you want to store the data.

Usage

```
gmCreateClass(name, type, description, parent="gmObject",
  envir=parent.frame(), db=getOption("gmDB"))
gmAddMembers(class, members, required, envir=parent.frame(), db=getOption("gmDB"))
gmCreateRightsGroup(rightsGroup, envir=parent.frame(), db=getOption("gmDB"))
gmCreateUserGroup(userGroup, envir=parent.frame(), db=getOption("gmDB"))
gmSetRights(userGroup, rightsGroup, write=TRUE, read=TRUE,
  envir=parent.frame(), db=getOption("gmDB"))
gmAddUserToGroup(user, userGroup, envir=parent.frame(), db=getOption("gmDB"))
gmSetRightsGroup(object, rightsGroup, envir=parent.frame(), db=getOption("gmDB"))
gmCreateUser(user, password, userGroups, envir=parent.frame(), db=getOption("gmDB"))
gmDefaultRightsGroup(rightsGroup, db=getOption("gmDB"))
```

Arguments

name	The name of the new class, which shall be created.
type	The type of the new class.
description	A short description of the calss and what it is for.
parent	The super class of the new one.
class	The name of the class to which the members shall be added.
members	The names of the new class members.
required	A logical. If TRUE this member has to be initialised, when an instance of the object is added to the database.
rightsGroup	The name or, if it refers an existing one, ID of the rights group.
userGroup	The name or, if it refers an existing one, ID of the usergroup
user	The username oder ID of the user, who shall be added.
write	A logical. If TRUE the stated usergroup is granted writing rights.
read	A logical. If TRUE the stated usergroup is granted reading rights.
object	An ID or gmExpr of the object, for that the rights group shall be determined.

password	The password of the new user.
userGroups	The usergroups the new user shall be part of.
envir	defines the environment used for evaluation of the other arguments.
db	A database connection object as returned by gmConnectServer.

Details

gmCreateClass allows the suer to add a new variable/class to the database. If set is given a type, the new class will be inferred from parent, which is gmObject by default.

gmAddMembers allows to add classes/variable to other classes. The option required determines, if this member must be initialised during construction of an instance of class.

gmCreateRightsGroup and gmCreateUserGroup create new rights or user groups. A rights group contains the sets readers and writers, consisting of usergroups, which determine, who is allowed to access or change an object. The rights group of an object can be set with gmSetRights after the initialisation of the object.

gmCreateUser is for creation of a new user. You may add this new user to the appropriate usergroups using gmAddUserToGroup.

During object initialisation using gmAdd the arguments rg can give the needed rights group. If is not given, a default value is used. This can be set by gmDefaultRightsGroup.

Value

gmCreateRightsGroup and gmCreateUserGroup return the gmID of this object.

Author(s)

S. Matos Camacho

See Also

gmConnectServer

Examples

```
## Not run:
gmConnectServer(MySQL(), host="myServer.mydomain.edu", user="mysqlUser",
  passwd="mysqlPassword", dbname="myDB")

rg = gmCreateRightsGroup("Protected")
ug = gmCreateUserGroup("Admins")
gmSetRights("Admins","Protected",read=TRUE,write=TRUE)
gmAddUserToGroup("admin","Admins")

gmSetRightsGroup(rg,"Protected")
gmSetRightsGroup(ug,"Protected")

rg = gmCreateRightsGroup("general")
ug = gmCreateUserGroup("users")
gmSetRightsGroup(c(rg,ug),"Protected")
```

```
gmAddUserToGroup("admin","users")
gmSetRights("users", "general", read=TRUE, write=TRUE)

gmCreateUser("myuser","mypassword",c("users"))
gmCreateUser(userGroups=c("users"))
gmDisconnectServer()

gmConnectDB(server="myServer.mydomain.edu",user="mysqlUser",passwd="mysqlPassword")
gmDefaultRightsGroup("general")

gmListVariables()

gmCreateClass("sample","set","a material sample",parent="gmObject")
gmAddMembers("root","sample", required=FALSE)
gmClassMembers(var="sample")
gmAdd(root,"sample",data.frame(gmName="Sample1"))

gmDisconnectServer()

## End(Not run)
```

---

gmExpr *Creating gmExpressions*

Description

gmExpr creates a gmExpression

Usage

```
gmExpr(expr,...,envir=parent.frame(),EXPR=substitute(expr))
```

Arguments

expr	an R-like representation of an SQL query.
envir	defines the environment used for evaluation of expr.
EXPR	The quoted version of such an expression.
...	further arguments for later use in other methods.

Details

gmExpr creates an object of class gmExpr, which can be used as argument in gmRead instead of a string. Furthermore it can be extended like any other expression used in gmRead. For use in the geometallurgical database you need to begin every expression with root, since data storage starts from this common root object.

The composition of an expression is done in the following way: for accessing a certain object in a class, use the $-operator, to access its elements use square brackets []. By specifying columnName=operation

the user can control the content of a column in the the output. Named columns are always shown in the result. If they need to be dropped set them to operation `drop`. If you want to group objects by a named column, set its operation to `group`. This grouping can only be done for elements at least one step down in the hierarchy.

Value

An object of class `gmExpr`, containing the expression in `$EXPR` and the `envir` as attribute.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

See Also

`gmRead`

Examples

```
expr <- gmExpr(root$project)
## Not run:
gmConnectServer(MySQL(), ...)
gmRead(expr$series)

gmRead(expr$series[name=gmTitle, name=drop])
## naming the element gmTitle as name and dropping it in the result

gmRead(expr[pID=id]$series[name=gmTitle, pID=group])
## grouping the series by their corresponding project in the result

## End(Not run)
```

---

gmGet *Get information about an object.*

Description

`gmGet` retrieves the information on specified objects stored in a geometallurgical database.

Usage

```
gmGet(expr,envir=parent.frame(),EXPR=substitute(expr),db=getOption("gmDB"))
gmGetVar(expr,what,EXPR=substitute(expr),envir=parent.frame(),db=getOption("gmDB"),
          unique=FALSE)
```

Arguments

expr	R expression of an SQL query. Expressions created with gmExpr can be used as well.
what	The members/variables you want the information from.
envir	defines the environment used for evaluation of expr.
EXPR	The quoted version of such an expression.
db	The database connection for the query.
unique	logical: If TRUE the execution is stopped in the case of multiple variables of the same requested variable name.

Details

Every object corresponding to the query is referred by its id. gmGet delivers a list consisting of a data frame info and another list sets. In info the values of all atomic members for every found object are stated. sets consists of data frames, called after the composite member variables, stating the object id and the id of the member object. Since these members are sets, it is possible that the data frames consist of several member objects for one referred object.

If you are interested in information on a certain variable but not the content of variables of this kind, use gmGetVar instead.

Value

list for gmGet, data.frame for gmGetVar

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

See Also

gmRead, gmExpr

Examples

```
## Not run: gmGet(root)
gmGetVar(root, "gmUser")
## End(Not run)
```

`gmGetVariableType` *Getting information on variables in a geometallurgy (gm)Database*

Description

Reads the variable name metainformation from the gmDatabase.

Usage

```
gmGetVariableType(sel,name,selID=attr(sel,"id"),db=getOption("gmDB"))
```

Arguments

`sel`	A call representing the intended SQL expression as R expression.
`name`	A character vector giving the names of the variables to be read.
`selID`	The ID attribute of the selection.
`db`	The database connection to the geometallurgical database.

Details

The geometallurgical database allows to store recursive dataframes of arbitrary content, provided that the same variable always contains the same type of information. This command allows to get the information on defined variables including their information.

Value

A dataframe with column name, table, decription and var, giving the name of the variable, the database table storing the information, the description (definition) of the variable, and the column of the database containing this data.

Author(s)

K. Gerald van den Boogaart

`gmJoinAVariable` *Creating the join R representation of a SQL JOIN*

Description

Queries the variable types of the variables, which shall be joined, and creates the R presentation of an SQL JOIN.

Usage

```
gmJoinAVariable(sel,var,nameUse=var,bind=TRUE,db=getOption("gmDB"),selID=attr(sel,"id"))
gmJoinADollarExpr(sel,expr,nameUse,envir,bind=TRUE,db=getOption("gmDB"),
                    gmNameVarID=getGmNameVarID(db))
```

Arguments

sel	The selection statement in R representation.
var	R expression of an SQL query.
nameUse	the names used for the variables
envir	defines the environment used for evaluation of expr
bind	should the variable be bound
db	The database connection to the material analysis database.
selID	The ID attribute of the selection.
expr	An R expression.
gmNameVarID	The gmVarID of gmName from the database.

Details

Internals functions for constructing a SQL statement.

Value

For gmJoinADollarExpression the new selection expression, where the dollar expression is added.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

---

gmJoinTheID *Add the ID to the selection*

Description

gmJoinTheID joins the ID attribute to the selection

Usage

```
gmJoinTheID(sel,nameUse,db=getOption("gmDB"),front=FALSE)
```

Arguments

sel	The selection statement in R representation.
nameUse	The name for the selection statement.
db	The database connection to the material analysis database.
front	A boolean determining if this ID shall become the first column in the later data frame. Default is FALSE.

Details

The variable nameUse will be used as name of the selection later referring. This function is intended for internal use in gmReadInternal only.

Value

The changed selection call.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

See Also

gmReadInternal

---

gmRead *Return the result of an SQl query given as R-like representation*

Description

gmRead translates the R representation expr of an SQL query into SQL and evaluates it in the database db.

Usage

```
gmRead(expr,envir=parent.frame(),EXPR=substitute(expr),limit=-1,db=getOption("gmDB"))
gmReadInternal(EXPR,sel=NULL,db,gmNameVarID=getGmNameVarID(db))
```

Arguments

expr	R expression of an SQL query. Expressions created with gmExpr can be used as well.
envir	defines the environment used for evaluation of expr.
EXPR	The quoted version of such an expression.
sel	The selection statement in R representation.
limit	defines the number of lines returned in the query.
db	defines the database, on which the query will be run.
gmNameVarID	The gmVarID in the gmDatabase of gmName for later referencing of an object by its name in the gmDatabase.

Details

gmRead is the principal function of the gmDatabase package. It provides the main functionality: Receiving an R expression expr of an SQL query, it returns a dataframe, containing the requested data. On the other hand, gmReadInternal, indicated by its notation, is meant for in-package use only. During the execution of gmRead it creates a call containing the R representation, which is used later in gmSQL to create and run a proper SQL query against the database.

Value

For gmRead a dataframe is returned. It contains the requested data. For gmReadInternal a call is returned, consisting of an expression to be interpreted as the R representation of SQL by gmSQL. Furthermore a list of the asked variables in EXPR and the columns, where to look in the database tables are returned, too.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

See Also

gmExpr, gmSQL

Examples

```
## Not run:
.gmDB <- dbConnect(...)
root <- "root"
erg <- gmRead(root)
## returns the ID of root in the database

gmRead(root$project[gmTitle=group])
## groups

## End(Not run)
```

---

gmRemove *Delete objects in an geometallurgical database*

Description

Deletes the specified object in an geometallurgical database.

Usage

```
gmRemove(expr, which=NULL, var=NULL, EXPR=substitute(expr), WHICH=substitute(which),
  envir=parent.frame(), db=getOption("gmDB"))
```

Arguments

expr	R expression of an SQL query. Expressions created with gmExpr and an ID in the database can be used as well.
which	Specifies for a set variable, which entry shall be deleted.
var	Defines the attributes, that shall be deleted.
EXPR	The quoted version of such an expression.
WHICH	The quoted version of which.
envir	defines the environment used for evaluation of expr.
db	The connection to the database.

Details

gmRemove deletes the specified object in a geometallurgical database. This may include the removal of all its members, too.

If you only want to remove a certain attribute for an object, than use the argument var for specifying. In the case of this attribute being an object itself but not a value, you may have more than one entry to choose for deleting. Therefor choose the right one by stating an id or an gmExpr in the which argument.

Value

No value.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

See Also

gmRead, gmExpr

Examples

```
## Not run:
gmRemove(4545)
gmRemove(root$project[gmTitle=="Project X"])
gmRemove(root$gmUserGroup, root$gmUser[gmUserName=="userXY"], var="gmUser")
gmRemove(root$project[gmTitle=="Project X"], var="gmTitle"))

## End(Not run)
```

---

gmSQL *Provide an R representation of SQL*

Description

gmSQL provides an R representation of SQL, which can be used to construct a complex hierachy of joins and select statements.

Usage

```
gmSQL(.,expr=substitute(.),env=SQLenv)
gmSQL2SQL(expr,env=SQL2SQLenv)
gmSQLTable(table,as=tick(table))
```

Arguments

.	For gmSQL an unquote expression to be interpreted as the R representation of SQL described under details.
expr	The quoted version of such an expression.
env	The environment holding the variables used in the expression.
table	a character string giving the name of the SQL table denoted.
as	the alias of the table in the SQL expression

Details

These commands allow to construct a representation of a small subset of SQL statements by R language objects. The following (derived) table value statements are supported

- join(x,y,on=NULL)Represent the x JOIN y ON on.
- leftjoin(x,y,on=NULL)Represent the x LEFT OUTER JOIN y ON on.
- select(what=NULL,from=NULL,where=NULL,as=tick()) Represents (SELECT w1=v1,... FROM from WHERE where) AS as, where what is a named list of the form list(w1=v1,...). If any of the terms is NULL it is logically omitted.
- table(table,as)Represents table AS as in a FROM clause.
- table$nameRepresents tableAlias.name anywhere in an SQL expression e.g. in what and where clauses of a SELECT.
- Call(fun,...)Represents fun(...) in SQL expressions.
- x==yRepresents x=y in SQL expressions.
- x!=yRepresents x!=y in SQL expressions.
- x<yRepresents x<y in SQL expressions.
- x>yRepresents x>y in SQL expressions.
- x<=yRepresents x<=y in SQL expressions.
- x>=yRepresents x>=y in SQL expressions.
- between(x,y,z)Represents x BETWEEN y AND z in SQL expressions.
- x+yRepresents x+y in SQL expressions.
- x-yRepresents x-y in SQL expressions.
- x*yRepresents x*y in SQL expressions.
- x/yRepresents x/y in SQL expressions.
- in(x,y)Represents x IN y in SQL expressions.
- "x %in% y"Substitute for in(x,y).
- &Represents x AND y in SQL expressions.
- |Represents x OR y in SQL expressions.
- !Represents NOT x in SQL expressions.
- ifelse(x,y,z)Represents IF x THEN y ELSE z in SQL expressions.
- .(x)Evaluates its argument in env, i.e. it is used to quote calculation, which should be executed in R rather than SQL.

- x %«% yRepresents `x << y` in SQL expressions (Left shift).
- x %»% yRepresents `x >> y` in SQL expressions (Right shift).
- xor(x,y)Represents `x XOR y` in SQL expressions.
- x%&%yRepresents `x & y` in SQL expressions (bitwise and).
- x%|%yRepresents `x | y` in SQL expressions (bitwise or).
- x%<==>%yRepresents `x<=>y` in SQL expressions (null safe equality).
- x%%yRepresents `x % y` in SQL expressions (remainder).
- x&&yRepresents `x && y` in SQL expressions (logical AND).
- x||yRepresents `x || y` in SQL expressions (logical OR).
- c(...)Represents `(...)` in SQL expressions.
- sum(x)Represents `SUM(x)` in SQL expressions (sum of values).
- avg(x)Represents `AVG(x)` in SQl expressions (average of values).
- min(...)Represents `MIN(...)` in SQL expressions (minimum of values).
- max(...)Represents `MAX(...)` in SQL expressions (maximum of values).
- count(x)Represents `COUNT(x)` in SQL expressions.

Special environments `SQLenv` and `SQL2SQLenv` are used in order to prevent code injection.

Value

For gmSQL and gmSQLTable, a call representing the intended SQL expression as R expression. For gmSQL2SQL a character string holding the corresponding SQL expression.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

References

http://dev.mysql.com/doc/refman/5.7/en

See Also

dbSendQuery

Examples

```
tabA <- gmSQLTable("A")
tabB <- gmSQLTable("B")
ennv <- new.env(parent=SQLenv)
assign("tabA", tabA, ennv)
assign("tabB", tabB, ennv)
AB <- gmSQL(join(tabA,tabB,on=tabA$id==tabB$refID), env=ennv)
AB
gmSQL2SQL(tabA)
gmSQL2SQL(tabB)
```

```
gmSQL2SQL(AB)
legalvalues <- 1:3
assign("AB", AB, ennv)
assign("legalvalues", legalvalues, ennv)
sAB <- gmSQL(select(what=list(x=1,y=tabA$y*tabB$y),
                    from=AB,
                    where=Call("log",tabB$othervalue)<=17 &&
                          IN(tabA$inte,c(legalvalues)),
                    as=NULL), env=ennv)

cat(gmSQL2SQL(sAB))
```

---

gmSQLValues *Format vector in parenthesis and SQL quote it*

Description

Formats values for use in SQL statements

Usage

```
gmSQLValues(v,quote=TRUE,db=getOption("gmDB"))
```

Arguments

v	The values to be converted, could be dataframe, numeric vector or something convertable to a character vector.
quote	Logocical value defining if the given values shall be quoted.
db	The database connection for which the transformation shall be done. Currently only mysql is supported.

Details

The functions currently use dbEscapeStrings and are therefor only reliable with RMySQL-Connections.

Value

gmSQLValues

A single string containing an SQL representation of the vector for use with the IN operator.

Author(s)

K. Gerald van den Boogaart

See Also

gmEscapeStrings

Examples

```
## Not run:
.gmDB <- dbConnect(...)
complexString <- "He said: \"I\'m going to the circus\"\n"
cat(complexString)
gmEscapeStrings(complexString)
cat(gmEscapeStrings(complexString),"\n")
cat(gmSQLValues(complexString),"\n")
myfactor <- factor(c("a","a","b"))
cat(gmSQLValues(myfactor),"\n")
num <- 1:3
cat(gmSQLValues(num),"\n")

X <- data.frame(string=rep(complexString,3),fac=myfactor,x=num)
cat(gmSQLValues(X),"\n")

## End(Not run)
```

---

makeGmSQLfromR *Parse an R expression of SQl statements*

Description

makeGmSQLfromR parses an R expression

Usage

```
makeGmSQLfromR(EXPR,vars,warn=FALSE)
```

Arguments

EXPR	R expression of an SQL query.
vars	A list of variables
warn	A logical defining the handling of a possible occuring warning.

Details

makeGmSQLfromR is an internal function, which is used for constructing WHERE clauses in SELECT statements or ON clauses in JOIN statements out of an R expression.

Value

Returns the names for the WHERE or ON clause.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

See Also

gmRead

`repairGmGrandChilds` *Updating second order inheritance.*

Description

Updates the table `gmGrandChilds` in a geometallurgical database.

Usage

```
repairGmGrandChilds(db=getOption("gmDB"))
```

Arguments

db	defines the database, on which the query will be run.

Details

The table `gmGrandChilds` is used for a fast and effectiv access to the hierarchical structure of a geometallurgical database. Here inheritance of second order is recorded. This information is necessary for retrieving all class members for a given object in `gmClassMembers`.

Value

string if any changes made in `gmGrandChilds`.

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

See Also

gmClass, gmClassMembers

replaceVarsInExpression *Replace Variables in Expressions*

Description

Replaces the variables in expressions by something else and returns the expression.

Usage

```
replaceVarsInExpression(EXPR,vars,warn=FALSE)
```

Arguments

EXPR	a call object representning an R expression
vars	a names list providing the replacements for the variables. The names represent the variable names to be replaced.
warn	If TRUE warns in case of undefined variables. NA produces an error.

Value

An expression similar to EXPR, with the variables replaced.

Author(s)

K. Gerald van den Boogaart

Examples

```
a <- quote( A==B^C+A^2+C )
replaceVarsInExpression(a,list(A=quote(2*r^2),C=as.name("R"),B=pi))
a <- quote( A+B+C )
replaceVarsInExpression(a,list(A=quote(C^2),B=quote(A^2),C=quote(B^2)))
```

tick *Create a unique name*

Description

tick creates a unique name out of x

Usage

```
tick(s="T")
```

Arguments

s	The first part of the later name.

Details

tick concatenates a consecutive number to s.

Value

a string

Author(s)

K. Gerald van den Boogaart, S. Matos Camacho

Examples

```
a <- "S"
tick(a)
```

Index