# MUTALL

## Info

In this document, and in others to follow, the OUTLOOK library uses the lowercase snake\_case naming convention to identify the:

files file\_paths

namespace class names. properties method names

Method arguments

This ensures consistency in the management of our codebase.

## Overview

NOTE: There are two naming conventions associated with the name mutall namely;-

1. mutall is the super class that contains commoon methods for all classes within the mutall project.
2. mutall is a namespace. Within the scope of this document, a prefix namespace mutall is used to associated claases and files.Therefore, in examples within this document and others, the reference

mutall\class\_name()

implies that the namespace **mutall** is referenced first and the class\_name associated with that namespace follows. In general, through this document and in associated/referenced documents, the example above denotes the convention of associating class names with their namespace reference.

## Class reference

Methods

### fetch

This method is executed in export.php whereby it is designed to support the execution of arbitrary methods and class objects from Javascript by constructing a structure {class\_name, cargs, method\_name, margs} which contains the class name, class arguements (constructor parameters), the method to execute, and the method parameters.

static

function

fetch

()

/\*:

result

\*/

The return value of this method is identical to the return value of the method identified by the **(method\_name)** during execution.

### get\_error

This method uses an (\Exception $ex) as its only parameter and returns/reports a string ofException(s) in a friendly manner that are otherwise unreadable in their unstructured form. From the Exception, this method reports a stacked trace of logged exception messages. This means that the execution of methods is collected and the method execution chain is recorded in the trace to support easier debugging and reporting of errors at runtime.

static

function

get\_error

(

\Exception $ex

)

:

string

The return value from this method is the stack trace as a well formatted string.

### post\_file

static

function

post\_file

()

:

bool

This method is responsible for moving uploaded files such as images from any location, to a designated

location within the server.

On successful transfer of the file, the method return

true

and

false

if the transfer was unsuccessful.

One critical file movement convention that applies here is that:-

If the uploaded file bears a signature name similar to an existing file, then the

**newly uploaded**

**file** overwrites the **existing file.**

The return values is true on successful execution of the method, and false when the execution was unsuccessful.

### ignore

static

function

ignore

()

:

void

This method ignores variables that are not used during the destructuring of position based elements.**NOT COMPLETELY DEVELOPED**.

**offload\_properties**

static

function

offload\_properties

(

array $dest, array $src

)

:

:

/\*

false

|

array

\*/

Uses a

destination

and a

source

as valid JSON arrays to offload the properties from the source to

the destination.

If the source is empty, then the method fails to execute and returns

false

which is one of the return

values of this method.

If the source is not empty, the properties are offloaded and the

destination

JSON array is generated.

This method returns a void or an array(Recheck the return type).

### save\_requests

Uses a stingified filename as the method argument to save the request generated during POSTing to a file. The format of the request is a JSON format for easier debugging.

static

function

save\_requests

(

string $filename

)

:

void

This method does not have any return values.

### save\_session

Creates a session for the logged in user, using the username and passqord to avoid passing these credentials every time we create the database object. These details do not change once the session is created.

static

function

save\_session

(

string: $username, string: $password

)

:

void

No return values are expected from this method.

### search\_class

Uses the class\_name to resolve user defined classes located in the schema\v\code\ directory by automatically loading the classes found in this directory. It assumes one file per class is saved in the root folder.

static

function

search\_class

(

string:$class\_name

)

:

void

In hind site, the method extracts the URL such as

http://localhost:90/tracker/v/code/index.php and extracts the filename associated with that

URL. The path is reconstructed with its full path, whether the path it is located on is a relative or a partial reference, including the document root to the location the file is located. This way any reference to a given class name, it reconstructed, and loaded.

**set\_requests**

static

function

set\_requests

(

string $filename

)

:

void

Gets the contents of the file that was saved as part of the request in

save\_requests()

and generates a

JSON string containing the request.

This method returns nothing.

Properties

By default, all properties in this class are public.

1. class\_name

A stringed name that characterizes every mutall object with this property and is relevant in the reconstruction of the class using the reflection class. The Reflection class serves the purpose of broadcasting information about a class. This way, all methods, properties, objects, and constants are loaded and interrogated for error reporting in advance.

public

string $class\_name

1. namespace

A string that contains the namespace identifying the current mutall object. From the Reflection class, the namespace is also extracted for easier association of classes and their contents. This namespace classification serves as a grouping of how classes objects, properties, methods, and constants are associated during execution and it improves the readability of source code which eliminates issues associated with method,property, or constants' name collision.

1. Exception determinant

A boolean exception determinant used to determine when to throw exceptions during the execution of methods or classes. By default the exception determinant is true bool $throw\_exception = true

Example

NOTEThis class is only accessible by extension namelye class class\_name extends mutall and it is a convention that will repeat atleast once.

# SCHEMA

## Overview

This class is suitable for modelling special mutall objects associated with the mutall database objects namely the database, entity, index, and column. -The above named database objects extend this class. This class extends the mutall class.

class

schema

extends

mutall

## Class Reference

### Methods

**Synopsis**

1. full\_name

public

function

full\_name

()

:

string

1. save

public

final

function

save

(

/\*row|null\*/

$row)

:

answer

1. loggin\_is\_necessary

public

function

logging\_is\_necessary

(

$row

)

:

true

**full\_name**

Generates a full name associated with a node and is generated from the class name and the partial name of the schema object. The combination is handled by a method because the partial name may change during runtime for example during barrel saving.

The method returns a string that represents the full\_name.

**save**

Exports a schema object to the database by:opening the save tag named by using the partial name. writing the a schema object to the database.

closing the save tag.

The final keyword associated with this method implies that the method cannot be overridden.

The method returns an answer. An answer is an expression that participates in the save operation.

Coming from a typescript definition of return types, it is easier to represent the answer with type

answer= myerror|scalar.

Expressing ourselves in PHP in this format is difficult and can be made possible by using the answer interface.

**logging\_is\_necessary**

### Constants

### Properties

1. partial\_name

public

string $partial\_name

The partial name is generated by the name of the schema object plus the database source it requires to formulate xml tags, indexing joints, constructing structurals, e.t.c.

1. roll\_back\_on\_fatal\_error

This toggles a default auto commit of transactions to false to influence a rollback when transactions are unsuccessful and commit upon the successful termination of transaction and returns true.

static

bool $roll\_back\_on\_fatal\_error=

false

By default, it is false, which implies that a transaction should not rollback incase there is an error at runtime.

1. errors

Schema objects take two forms, namely a static and an activated form. When a schema object is activated , any generated errors are managed by this property. 4. answer

public

array

/\*error[]\*/

$errors=[]

public

/\*\*/

$answer

After a schema object is written the database, the result is stored in this property.

### Interfaces

1. answer

interface

answer

{

function

\_\_toString

()

;

function

get\_position

()

/\*:

row\_index

,

col\_index

?]|

null

\*/

;

}

An answer is an expression that can actively participate as an output expression from a save operation. From a Typescript type definition, the answer can be specified as type answer= myerror|scalar. Using PHP, expression union return values is not directly possible, but can be done by using interfaces.

a. toString

function

\_\_toString

()

;

Generates an answer that must be converted to a string to enable creation of xml-like logs.

b. get\_position

function

get\_position

()

/\*[

row\_index

,

col\_index

?]|

null

\*/

;

Gets the position data associated with an answer to enrich an answer with position data when its available. Data can be recovered using this function as well

### User

### Contributions

### Example

# Database

## Overview

This is the class that is used to model the database objects.

One of the objects it models is the schema object that allows saving to multiple database. The other object is the database object that models the database ready for CRUD operations to be performed on the modelled database.

## Class Reference

### Methods

### Synopsis

1. authenticate

public

function

authenticate

(

string $name, string $password

)

:

bool

1. register

public

function

register

(

string $email, string $password

)

:

void

1. chk

public

function

chk

(

string $sql

)

:

string

1. get\_sql\_data

public

function

get\_sql\_data

(

string $sql

)

:

array

1. accounting

public

function

accounting

(

string $account\_name

)

:

array

1. export\_structure

public

function

export\_structure

()

:

database

;

1. query

public

function

query

()

:

int

;

1. lastInsertId

public

function

lastInsertId

()

:

int

;

### authenticate

Used to check whether a user with the given username $name and password $password credentials is found in the database or not.

An error is throw when more than name asscociated with the user is found.

When the user with the defined credentials :-

Is FOUND in the database, the return value of true is observed. Is NOT FOUND in the database, the return value of false is observed.

### register

Create a new account for a user using the provided username and password.

If :-

NO USER is found in the database,the user is created and linked to the entity used to hold all user related information which minimizes errors that may occur from double registration of the user.

THE USER EXISTS WITHOUT A PASSWORD, the password associated with the user's name is saved.

A USER EXISTS with the provided credentials, an Exception indicating that the user with the specified email already exists is provided.

This method does not return any values.

### chk

Check if an sql query is valid or conforms to the database specified querying standards depending on the database engine (mysql, postgres,oracle,mssql,or sqlite)selected.

Suitable for debugging a query before executing it to prevent runtime errors.

If the query provided is invalid, an Exception is thrown with the message associated with the violation/error displayed before the query string.

The return value is the string if the query provided had no errors.

### get\_sql\_data

Execute a query/sql provided as the only argument and return an array of all records associated with that result.

If there:-

Is an error associated with the sql, an exception is thrown with the exception message appearing before the query itself.

Is no error associated with the sql, an array of the results is returned.

### accounting

Retrieve all the accounting information associated with a specified account saved in the database. The return value of this method is an array with information such as the date,ref\_num, purpose,debit , and credit associated with the provided account name.

### export\_structure

Exports the entire database structure as it is defined by the PDO, with all its entities and columns.

The return value of export\_structure is a database object which can be used to reconstruct the database using its entities on the client-side platform(s).

### query

Extracts the number of affected rows associated with the query passed as the only argument.Note that this method only returns the number of rows and not the results obtained from the passed query.

The return value is an integer of the number of rows.

### lastInsertId

Extract the Id/primary key of the last inserted record in the database.

The return value is a string/number when there is value that was previously inserted.

As the name dictates, its applicability is limited to the Insertion process.

### Constants

### Properties

1. entities

This array is the collection of the tables needed to create the database. It is initialized as an empty array by default and it is populated later when the database object is created. 2. pdo\*\*(in the lastInsertId method, it is stated to be protected)

public

array

$entities= [];

public

\PDO $pdo;

Allows users to instantiate a database using this class to query and retrieve information from the

database.

All database objects constructed within this class originate from the

PHP Data Objects(PDO)

.

It is initialized as a property to avoid the class from extending a PDO object.

3

.

report

public

string $report;

This is where any errors that are reported are saved.

### Interfaces

### User

### Contributions

### Example

Modelling any database that requires the creation of a database object before making queries to the database. To use the database's object, three parameters are required namely; A mandatory stringified name of the database.

A boolean value to determine whether you want the complete database object with all its entities revealed.

The default value is true to show the complete database structure.

A boolean value to determine whether to throw an exception of database related errors as soon as they are encountered.

The default value is

true

.

Some of these errors may include indexing errors in the database.

$dbase =

new

database(

"mutall"

)

-

If no database name is provided, then a database name is selected by default.

Ensure that the database to prevent errors from occurring during database

modelling.

1. authenticate
2. register
3. chk
4. get\_sql\_data
5. query
6. accounting
7. export\_structure
8. query

# Entity

## Overview

This class represents an entity. By definition, an entity is a schema object which implies that it can be saved to a database.

## Class Reference

### Methods

### Synopsis

1. toString

abstract

function

\_\_toString

()

:

string

{}

1. yield\_entity

abstract

function

yield\_entity

()

:

\

Generator

{}

1. yield\_attribute

abstract

function

yield\_attribute

()

:

\

Generator

{}

1. foreigners

abstract

function

foreigners

()

:

\

Generator

{}

1. structural\_foreigners

abstract

function

structural\_foreigners

()

:

\

Generator

{}

1. structural\_pointers

abstract

function

structural\_pointers

()

:

\

Generator

{}

1. get\_id\_columns

abstract

function

get\_id\_columns

()

:

array

{}

### toString

Compiles the complete stringified version associated with this entity as an sql object for use in array methods.

The return value is a string made up of the database name and the entity name.

### yield\_entity

An entity yields itself in the form of an iterable array of entities in the form of a \Generator.

### yield\_attribute

Generates the attributes associated with an entity which are also based on its columns. The attributes are generated in a for each column.

This method returns an iterable array of attributes in the form of a \Generator.

### foreigners

Return the foreign key columns associated with this entity.

Pointers to the foreign key columns are excluded because they take time to build and may not always be required when extracting the foreign keys.

The result of this function should not be bufferred because with the addition of views into our model, the structure of the database can change at runtime.

The return value is an array of foreigners wrapped in the \Generator.

### structural\_foreigners

Structural foreigners are foreign keys that are not used for reporting purposes. This method iterates through all columns associated with this entity and extracts the structural foreigners.

The return value is an array of structural foreigners in the \Generator; **structural\_pointers**

Yield the structural pointers associated in the current database using the structural foreign keys. The generated foreigner is a marked as a pointer if its database are the same and the table names are the same.

The return value is an array of pointers enclosed in the \Generator.

### get\_id\_columns

Extract the mandatory columns associated with this entity, the columns that are not nullable

i.e. those that are not defined as NULL, and those used as ids as a record that needs to be saved.

The return value is an array of the mandatory columns.

Constants

Properties

1

.

name

public

string $name;

The name assigned/associated with an entity.

2

.

columns

public

array

$columns;

The columns of the entity.

3

.

indices

public

array

$indices;

The indices of this entity that are initialized during the construction of the database object.

### Interfaces

### User

### Contributions

### Example

# Table

## Overview

## Class Reference

This class models the actual table in the database by extending the entity and this way it includes the indexes as a dependency.

Methods

### Synopsis

1. pointers

public

function

pointers

()

:

\

Generator

{}

1. get\_friend

public

function

get\_friend

(

int $pk

)

:

string

{}

1. get\_friendly\_name

public

function

get\_friendly\_name

(

int $pk

)

:

string

{}

1. get\_dependency **pointers**

Yields an array of pointers as all the foreigners that reference this table and associated with the current database.

This method is similar to foreigners() with the only difference being that its output cannot be buffered, because with the ability to add a view to the database, the pointers to an entity can change.

The return value is an array of pointers.

### get\_friend

Retrieve the friendly component of the primary key passed as the only argument to this method. If no friendly component is defined or associated with the provided primary key, an Exception is flagged to note that none of the identified columns are associated with this `primary key.

The return value is a string which of the friendly component/column. The column may be indexed by the column name.

### get\_friendly\_name

Extracts and returns the friendly name of the given primary key.

If no friendly name is obtained, an exception is thrown with the message Invalid friendly name result ..... This message means that there is no friendly name associated with this primary key.

The return value is a string describing the friendly name of the primary key.

Constants

Properties

1

.

dbname

public

string $dbname;

The database name in which this table is contained.

2

.

depth

public

?int $depth=

null

;

The relation depth of this entity. It is set to null by default.

3

.

comment

public

$comment;

The

JSON

user information retrieved from the comment after it is decoded.

4

.

indices

public

array

$indices=[];

The unique indices of this table/entity that is used for identification of a record in a table. It is set

before being used.

By default, this index is empty and is

updated during database creation when the

indices created are valid

.

### Interfaces

### User

### Contributions

### Example

# view

## Overview

This class models the sql's of type select that extend an entity, so that they can take part in the database modelling exercise to resolve the root entity that requires the inclusion of a config file in the main application.

This class cannot be instantiated from javascript because the data types are not that simple.

## Class Reference

### Methods

### Synopsis

1. id

public

fuction id():string{}

1. yield\_entity

public

function

yield\_entity

()

:

\

Generator

{}

1. yield\_attribute

public

function

yield\_attribute

()

:

\

Generator

{}

1. execute

public

function

execute

()

/\*:

value

[][

cname

]

s

\*/

{}

1. stmt

public

function

stmt

()

:

string

{}

1. to\_str

public

function

to\_str

()

:

string

{}

### id

The short form of identifying a view which is a concatenated string of the database name and the entity name.

The return value is a string of the database and entity name.

### yield\_entity

Yields the trivial entity in this view and includes all the target entities that are involved in this join. **yield\_attribute**

Yields the columns that are involved in this view and are useful for editing a non trivial view(sql). **execute**

Executes the sql or query to return the data as a double array. It is assumed that at this point all the view constructor variables are set to their desired values. This is clearly not true for extensions like editor and selector. They must override this method to prepare the variables before calling this method.

The return value of this method is a double array.

### stmt

Extracts the string representation of a select sql statement.

The return value is a string containing the sql.

### to\_str

Compiles the scalar value associated with an sql statement.

The return values is a string of the scalar value extracted from the sql statement.

### Constants

### Properties

1

.

where

public

?expression $where;

Defines the criteria of extracting information from an entity as a boolean expression.

2

.

from

public

entity $from;

This

from

clause identifies the origin of this view which is the

entity

.

3

.

join

public

?join $join;

This view has connections on the various entities that are involved in this sql.

4

.

group\_by

public

group\_by= [];

Other clauses associated with an sql that the user can provide after a view is created.

### Interfaces

1. expression

interface

expression

{

//

//

function

to\_str

()

:

string

;

//

function

yield\_entity

()

:

\

Generator

;

//

function

yield\_attribute

()

:

\

Generator

;

}

-This general form of an expression was originally designed to support database querying operations function to\_str():string.

Every expression must be a valid sql expression.

In most cases, this method returns the same value as the \_toString() magic method but in some cases, it does not. For example the \_\_toString() of the if field in a selector such as the mutall.application.id\_\_ whereas its to\_str() value is concat(mutall\_login.application.name,"/").

Therefore, the \_\_toString() of an application entity is e.g.

mutall\_login.applicationbut that of the application expression, to\_str() refers to the primary key fieldmutall\_login.application.application.

function yield\_entity(): \Generator

Yield the entities that participate in this expression.

It is important for defining search paths for partial and save\_indirect view. This is the method that makes it possible to analyse the mutall view and do things that would currently not be possible without parsing sql statements.

function yield\_attribute():\Generator

Yield the primary attributes that are used in formulating this expression.

It is important for determining if a view column is directly editable or not.

It also makes it possible to express values by accessing the primary entities that constitute them.

### User

### Contributions Example

# column

## Overview

This class is used to model columns of an entity as the smallest package that can be saved to the

database using an expression.

This class implements the expression interface.

NOTE: This class is not abstract as we can use it to create columns. This is

important in the context of left joins.

## Class Reference

### Methods

### Synopsis

1. get\_error\_report

public

function

get\_error\_report

(

int $no\_of\_errors, string

&$report)

:

void

{}

1. yield\_attribute

public

function

yield\_attribute

()

:

\

Generator

{}

1. yield\_entity

public

function

yield\_entity

()

:

\

Generator

{}

### get\_error\_report

Returns the error report and the number of errors obtained at the database level.

There is no return value for this method.

### yield\_attribute

yields an attribute column since all other columns cannot yield their own column NOTE: This method is not yet fully developed/documented..

### yield\_entity

Yields the entity of this column or the entity associated with this column.

The return value is an entity.

### Constants

### Properties

1. name

public

string $name;

2 ename

public string $ename;

### Interfaces

### User

### Contributions

### Example

# capture

## Overview

This class models the primary columns as opposed to the derived columns needed for data capture and storage operations.

The columns are extracted from the information schema directly and therefore, they should be

checked for integrity

.

abstract

class

capture

extends

column

{

}

It extends the column class and it is abstract. This means that other classes can only implement the

methods in this class.

## Class Reference

### Methods

### Synopsis

1. [\_\_toString](# \_\_toString)

abstract

function

\_\_toString

()

:

string

{}

1. is\_cross\_member

abstract

function

is\_cross\_member

()

:

void

{}

3 is\_id

abstract function is\_id():bool{}

4. is\_descriptive

abstract

function

is\_descriptive

()

:

bool

{}

### \_\_toString

Extracts a stringified version of a capture column which is the same as that of an ordinary column prefixed with the database name to take care of scenarios where we have multiple databases.

The return value is a string.NOT ENFORCED IN THE CODE.

### is\_cross\_member

Extracts the non\_structural columns of this entity (cross members).

These are the optional foreign key columns, namely those that are not nullable.

They are important for avoiding cyclic loops during saving of data to the database.

This method does not return a value.

### is\_id

Evaluated whether this column is used by any identification index. Identification columns are part of the structural columns.

The return value of this method is true if this column is used for an identification index and false if it is not used.

### is\_descriptive

Finds out whether this column can be used for formulating a friendly identifier.

The return value is true if this column is used in the formulation.

### Constants

### Properties

1

.

dbname

public

string $dbname;

The name of the

current database

to support global access to the database and entity names.

NOTE:Constructing the details of a column requires the following;

2 comment

public ?string $comment;

Acts as a container that stores the metadata of this column as a structure if the metadata is not offloaded since we need to acquire it in its original form.

The comment can also be optional since it is set to be null as denoted by ?string $comment. 3. default

public

?string $default;

The database default value for this column.

The default can also be optional.

4

.

data\_type

public

string $data\_type;

The acceptable datatype for this column such as the

text, number, auto-number etc

s.

5

.

is\_nullable

public

string $is\_nullable;

This is defined if this column is mandatory or not.

A stringified value of

"YES"

if not nullable or a stringified value of

"NO"

if it is not nullable.

6

.

length

public

?int length;

The size of the column.

\*\*\*

NOTE:This value is optional.

7

.

type

public

string $type;

The type of the column. This is needed for the extraction of enumerated choices.

### Interfaces

### User

### Contributions

### Example

# Database

## Overview

Primary and foreign key columns are used for establishing relationships with the entities during data capture.

class

primary

extends

capture

{

}

The primary key column:-

Is named the same as the entity where it is located.

It has the

autonumber

datatype or the

int

that is an

AUTOINCREMENT

.

It extends the capture class.

## Class Reference

### Methods

### Synopsis

1. yield\_entity

public

function

yield\_entity

()

:

\

Generator

{}

### yield\_entity

Yield the attribute of an entity from a database that is opened.

The return value is a the name of the entity wrapped up in the \Generator.

### Constants

### Properties

### Interfaces

### User

### Contributions

### Example

# attribute

## Overview

This class creates/models attributes with its structure components as defined in the capture above.

Attributes have very special columns that have options that describe the data they hold such as

the

data type

,

their lengths

.

These descriptions are not owned by other columns.

class

attribute

extends

capture

implements

expression

{

}

This class extends the capture class and implements the expression interface.

## Class Reference

### Methods

### Synopsis

1. yield\_attribute

public

function

yield\_attribute

()

:

\

Generator

{}

### yield\_attribute

Yields this(current) attribute.

### Constants

### Properties

### Interfaces

### User

### Contributions

### Example

# field

## Overview

This class is used for modelling the derived columns, i.e., columns that are not read off the information schema.

class

field

extends

column

implements

expression

{

}

It extends the

column class

and implements the

expression

interface.

## Class Reference

### Methods

### Synopsis

1. \_\_toString

public

function

\_\_toString

()

:

string

{

//

}

### \_\_toString

Overrides the PHP magic constant \_\_toString to ensure that the dbname is not included as part of the name of the field.

The return value is a string that is usable in the database to associated this entity to its name.

### Constants

### Properties

1

.

exp

public

expression $exp;

The calculated/derived expression represented by this field.

### Interfaces

### User

### Contributions

### Example

# foreign

## Overview

The foreign class models the foreign keys columns in the database.

A foreign key column participates in data capture.

It implements the many-to-one-link between 2 entities:-

The first one is the

home entity

that houses the column

The second one is the

away entity

and it is the one pointed to by the relationship.

class

foreign

extends

capture

implements

ilink

{

}

This class extends the capture column and implements the ilink interface.

## Class Reference

### Methods

### Synopsis

1. verify\_integrity

public

function

verify\_integrity

()

:

void

{

}

1. away

public

function

verify\_integrity

()

:

void

{

}

1. home

public

function

home

()

:

entity

{

}

1. yield\_entity

public

function

yield\_entity

()

:

\

Generator

{

}

1. is\_hierarchical

public

function

is\_hierarchical

()

:

bool

{

}

### verify\_integrity

A foreign (key) must satisfy the following conditions to be compliant with the framework standards:-

1. The data type of the foreigner must of int.
2. The referenced column name must be a primary key.

If the foreign is not an integer, an error is thrown to show that the datatype of the column (foreign) is not of type int.

If the referenced column is not a primary key, an Error is thrown which dictates that the foreign key column should reference the reference a table using the primary key.

There is no return value for this method.

### away

Retrieves and returns the entity that the foreign key is pointing at the referenced entity.

### home

Extracts and returns the entity in which the foreign key is housed.

Within the data model, it is indicated with a chicken foot NOTE: data models are interpreted differently from class models.

### yield\_entity

Yield the entity associated with this column.

The return value of this column is attached to the generator function.

### is\_hierarchical

Tests whether a foreign key is hierarchical or not.

It is marked as hierarchical if:the table it references is the same as that of the parent it exists in the same database.

### Constants

### Properties

1

.

ref

public

\stdClass

/\*{ref\_table\_name, ref\_db\_name}\*/

$ref;

The name of the referenced table and database as an object.

Interfaces

1

.ilink

The link interface allows the expression of relationships between 2 columns as a string.

interface

ilink

{

function

on\_str

:

string

;

}

A link implements the

on\_str

needed to model joins in a join clause such as strings for the:-

many-to-one case.

one-to-one case.

### User

### **Contributions**

### Example

# pointer

## Overview

This class models a column in an entity that points to a referenced entity.

The difference between a pointer and a foreign is that the pointer is not home at the entity it is found.

class

pointer

extends

foreign

{

}

This class extends the

foreign

class.

## Class Reference

### Methods

### Synopsis

1. away

public

function

away

()

:

entity

{

}

1. home

public

function

home

()

:

entity

{

}

1. to\_str

public

function

to\_str

()

:

string

{

}

### away

Extracts and returns the home of the referenced entity.

pointers run in the opposite direction to corresponding foreign keys and so that its away entity is the home version of its foreign key.

### home

Extracts and returns the referenced entity.

Since pointers run in the opposite direction of the corresponding foreign keys, the home entity is the away entity of its foreign key.

### to\_str

Compiles and returns an expression string version of a column which is the concatenation of the dbname, ename, and cname.

The return value is the concatenated string.

### Constants

### Properties

### Interfaces

### User

### Contributions

### Example