

---

# **link.dbrequest Documentation**

***Release 0.19***

**David Delassus**

September 02, 2016



<b>1</b>	<b>Installation</b>	<b>3</b>
<b>2</b>	<b>Contents</b>	<b>5</b>
2.1	Examples . . . . .	5
2.2	API documentation . . . . .	6
<b>3</b>	<b>Donating</b>	<b>17</b>
	<b>Python Module Index</b>	<b>19</b>



**link.dbrequests** is a database agnostic query system.

Check out the source code on [Github](#).



---

## Installation

---

```
pip install link.dbrequests
```





---

## Contents

---

## 2.1 Examples

### 2.1.1 Getting a backend

```
from link.middleware.core import Middleware

# Will open a QueryManager using a MongoDB backend
manager = Middleware.get_middleware_by_uri('query+mongo://localhost:27107/mydatabase/mycollection')
# Will open a QueryManager using a SQL backend
manager = Middleware.get_middleware_by_uri('query+sql://localhost:5432/mydatabase/mytable')
```

### 2.1.2 Operations on the backend

```
from link.dbrequest.expression import E, F
from link.dbrequest.assignment import A
from link.dbrequest.comparison import C

query = manager.all() # get an iterable over all elements

manager.create(A('foo', 'bar')) # put document {'foo': 'bar'} into database

doc = manager.get(C('foo') != 'bar') # get single element, or None
```

### 2.1.3 Operations on queries

```
docs = list(query) # iterate over query to execute the request
docs = list(query) # use cache when iterating again

# create a new query from the first one
q2 = query.filter(C('foo') == 'bar')
assert query is not q2

# exclude documents without a field named "bar"
q3 = q2.exclude(~C('bar'))

# filter documents "weight > 5" and "prop1 < prop2 * 5"
```

```
q4 = q3.filter((C('weight') > 5) & (C('prop1') < (E('prop2') * 5)))

# set "prop3 = prop1 + prop2" on q2 result
docs = q2.update(A('prop3', E('prop1') + E('prop2')))

# delete documents
q3.delete()
```

## 2.1.4 Operations on documents

```
# save/delete a single document
doc.save()
doc.delete()
```

## 2.2 API documentation

### 2.2.1 link.dbrequest package

#### Submodules

#### link.dbrequest.manager module

**class** link.dbrequest.manager.**QueryManager** (*user=None, pwd=None, hosts=None, path=None, fragment='; \*\*kwargs*)

Bases: link.middleware.core.Middleware

Manage storage backend and provide query system for it.

**Parameters** **backend** (*Driver*) – Storage backend

**all** ()

Get a query selecting all elements in store.

**Return type** *Query*

**create** (*\*fields*)

Put a new element into the store.

**Parameters** **fields** (*list of A*) – List of assignments

**Returns** Created element

**Return type** *Model*

**execute** (*ast, scope=None*)

Send query to the storage driver.

**Parameters**

- **ast** (*dict or list*) – AST describing the query
- **scope** (*list of str or Query*) – Query's scope

**Returns** storage driver's response

**from\_ast** (*ast*)

Create a query from the provided AST.

**Parameters** `ast` (*list*) – AST

**Return type** *Query*

**Raises** **ASTError** – if provided AST is not valid.

**get** (*condition*)

Get a single element matching the filter.

**Parameters** `condition` (*C or CombinedCondition*) – Filter

**Returns** Matching element or None

**Return type** Model or None

**prepare** (*f, \*queries*)

Get a lazy procedure.

**Parameters**

- `f` (*F*) – Function to call
- `queries` (*list of Query or str*) – Function's scope

**Returns** New lazy procedure

**Return type** *Procedure*

**run** (*f, \*queries*)

Execute a procedure.

**Parameters**

- `f` (*F*) – Function to call
- `queries` (*list of Query or str*) – Function's scope

**Returns** Procedure's result

**Return type** any

**subset** (*scope*)

Get a query selecting all scoped elements in store.

**Parameters** `scope` (*str, Query or list*) – Query's scope

**Return type** *Query*

**static validate\_ast** (*ast*)

Validate AST semantics.

**Raises** **ASTError** – if the AST semantics are not valid

## link.dbrequest.driver module

**class** `link.dbrequest.driver.Driver` (*obj, \*args, \*\*kwargs*)

Bases: `link.feature.core.Feature`

Abstract storage driver feature.

**QUERY\_COUNT** = 'count'

**QUERY\_CREATE** = 'save'

**QUERY\_DELETE** = 'delete'

**QUERY\_READ** = 'find'

**QUERY\_RUN** = 'run'

**QUERY\_UPDATE** = 'update'

**count\_elements** (*ast*)  
Count number of elements matching the query described by the AST.

**Parameters** **ast** (*list or AST*) – AST describing the query

**Returns** number of elements matching the query

**Return type** int

**cursor\_class**  
alias of `Cursor`

**find\_elements** (*ast*)  
Find elements matching the query described by the AST.

**Parameters** **ast** (*list or AST*) – AST describing the query

**Returns** `Cursor` on matching elements

**Return type** *Cursor*

**model\_class**  
alias of `Model`

**name** = 'query'

**process\_query** (*query*)  
This method must be overridden, handles every query made to the storage.

**Parameters** **query** (*AST*) – query to process

**Returns** driver's response.

**put\_element** (*ast*)  
Put element into the store.

**Parameters** **ast** (*list*) – AST describing the element to insert

**Returns** Inserted element

**Return type** *Model*

**remove\_elements** (*ast*)  
Delete elements matching the query described by the AST.

**Parameters** **ast** (*list or AST*) – AST describing the query

**Returns** Number of elements deleted

**Return type** int

**run\_procedure** (*ast*)  
Run procedure on middleware.

**Parameters** **ast** (*AST*) – AST describing the query

**Returns** Procedure's result

**Return type** any

**update\_elements** (*filter\_ast, update\_ast*)  
Update elements matching the query described by the AST.

**Parameters**

- **filter\_ast** (*list*) – AST describing the query
- **update\_ast** (*list*) – AST describing the update

**Returns** Number of elements modified

**Return type** `int`

## link.dbrequest.model module

**class** `link.dbrequest.model.Cursor` (*driver*, *cursor*, \**args*, \*\**kwargs*)

Bases: `six.Iterator`

Cursor encapsulating storage driver's cursor or result.

### Parameters

- **driver** (`Driver`) – storage driver
- **cursor** – storage's cursor

**cursor**

**driver**

**to\_model** (*doc*)

Convert raw element to Model.

**Parameters** **doc** (*dict*) – raw element

**Returns** Element as Model

**Return type** `Model`

**class** `link.dbrequest.model.Model` (*driver*, *data*, \**args*, \*\**kwargs*)

Bases: `object`

Model class encapsulating elements in store.

### Parameters

- **driver** (`Driver`) – storage driver
- **data** (*dict*) – encapsulated data

**data**

**delete** ()

Delete element from store.

**driver**

**save** ()

Save element into store.

**Returns** Inserted document

**Return type** `Model`

## link.dbrequest.lazy module

**class** `link.dbrequest.lazy.Dataset` (*manager*, *scope=None*, \**args*, \*\**kwargs*)

Bases: `link.dbrequest.lazy.Lazy`

Lazy dataset, hold scoped set of data.

**class** `link.dbrequest.lazy.Lazy` (*manager*, *scope=None*, *\*args*, *\*\*kwargs*)

Bases: `object`

Object holding data that will be fetched by the manager when needed.

**Parameters**

- **manager** (`QueryManager`) – Manager that will execute this query
- **scope** (*str*, *Query* or *list*) – Object’s scope (default: `None`)

**execute** ()

Execute object

**Returns** Data held by object

**Return type** `any`

**class** `link.dbrequest.lazy.Procedure` (*f*, *\*args*, *\*\*kwargs*)

Bases: `link.dbrequest.lazy.Dataset`

**copy** ()

**execute** ()

**to\_ast** ()

**class** `link.dbrequest.lazy.Query` (*\*args*, *\*\*kwargs*)

Bases: `link.dbrequest.lazy.Dataset`

Database agnostic query.

**Parameters** **manager** (`QueryManager`) – Manager that will execute this query

**copy** ()

Create a copy of this query.

**Returns** `Copy`

**Return type** `Query`

**count** ()

Count elements matched by this query.

**Returns** Number of matching elements

**Return type** `int`

**delete** ()

Delete elements matching the query.

**Returns** Number of deleted elements.

**Return type** `int`

**exclude** (*condition*)

Returns a new query with a new (negated) filter added.

**Parameters** **condition** (*C* or *CombinedCondition*) – Filter

**Returns** `Query`

**Return type** `Query`

**execute** ()

**filter** (*condition*)

Returns a new query with a new filter added.

**Parameters** `condition` (*C or CombinedCondition*) – Filter

**Returns** Query

**Return type** *Query*

**get** (*condition*)

Add filter to the query, and get a single element matching the query.

**Parameters** `condition` (*C or CombinedCondition*) – Filter

**Returns** Matching element or None

**Return type** Model or None

**group** (*key, \*expressions*)

Group elements by key matching the query using the expressions.

**Parameters**

- **key** (*str*) – Key used to group elements
- **expressions** (*list of CombinableExpression*) – Expressions

**Returns** Grouped elements.

**to\_ast** ()

Returns a copy of query's AST

**Returns** AST

**Return type** list

**update** (*\*fields*)

Update elements matching the query.

**Parameters** `fields` (*list of A*) – Assignments

**Returns** Number of modified elements.

**Return type** int

## link.dbrequest.ast module

**class** `link.dbrequest.ast.AST` (*name, val, \*args, \*\*kwargs*)

Bases: dict

Helper for create AST nodes.

**Parameters**

- **name** (*str*) – AST node name
- **val** – AST node value

**name**

**val**

**exception** `link.dbrequest.ast.ASTError`

Bases: `exceptions.Exception`

Basic AST semantic error.

**exception** `link.dbrequest.ast.ASTInvalidFormatError`

Bases: `link.dbrequest.ast.ASTError`

Error raised when supplied AST is not a valid expected type.

**exception** `link.dbrequest.ast.ASTInvalidStatementError` (*stmt*)

Bases: `link.dbrequest.ast.ASTError`

Error raised when trying to execute an unknown statement.

**exception** `link.dbrequest.ast.ASTLastStatementError` (*stmt*, *pos*)

Bases: `link.dbrequest.ast.ASTError`

Error raised when trying to execute a statement that must be in the end of the sequence.

**exception** `link.dbrequest.ast.ASTSingleStatementError` (*stmt*)

Bases: `link.dbrequest.ast.ASTError`

Error raised when trying to execute statements that must be in a sequence.

**class** `link.dbrequest.ast.ModelBuilder` (*\*args*, *\*\*kwargs*)

Bases: `object`

Class used to link nodes in AST together.

**parse** (*node*, *parent=None*)

Parse AST to resolve link between nodes.

**Parameters**

- **node** (*AST or list or value*) – node to parse
- **parent** (*AST or None*) – parent node (default: `None`)

**Returns** `Node`

**Return type** same as param node

**class** `link.dbrequest.ast.NodeWalker`

Bases: `object`

Walk through AST.

**find\_walker** (*node*)

Find method used to walk through specific node.

**Parameters** **node** (*AST*) – node to walk through

**Returns** Method used to walk through the node

**Return type** callable or `None`

**walk** (*node*, *\*args*, *\*\*kwargs*)

Walk through AST (depth first).

**Parameters**

- **node** (*any*) – Root node to start walking through
- **args** (*iterable*) – Positional arguments for walker method
- **kwargs** (*dict*) – Keyword arguments for walker method

**Returns** Walker method's result



## link.dbrequest.tree module

**class** link.dbrequest.tree.**Node**(*name*, \**args*, \*\**kwargs*)

Bases: object

Base class for AST nodes.

**Parameters** *name* – Node’s value.

**get\_ast**()

Returns simplified AST.

**Returns** AST node as dict

**Return type** dict

**class** link.dbrequest.tree.**Value**(*name*, \**args*, \*\**kwargs*)

Bases: `link.dbrequest.tree.Node`

Node representing a value.

**get\_ast**()

## link.dbrequest.comparison module

**class** link.dbrequest.comparison.**C**(*name*, \**args*, \*\**kwargs*)

Bases: `link.dbrequest.tree.Node`, `link.dbrequest.comparison.Comparable`,  
`link.dbrequest.comparison.CombinableCondition`

Condition on a property.

**get\_ast**()

**class** link.dbrequest.comparison.**CombinableCondition**

Bases: object

Combine conditions with boolean operators.

**AND** = ‘and’

**OR** = ‘or’

**XOR** = ‘xor’

**class** link.dbrequest.comparison.**CombinedCondition**(*left*, *operator*, *right*, *inverted*=False,  
\**args*, \*\**kwargs*)

Bases: `link.dbrequest.tree.Node`, `link.dbrequest.comparison.CombinableCondition`

Combination of two conditions.

**Parameters**

- **left** (`CombinableCondition`) – left condition
- **operator** (*str*) – combination condition
- **right** (`CombinableCondition`) – right condition
- **inverted** (*bool*) – Equivalent of not (<condition>) (default: False)

**get\_ast**()

**class** link.dbrequest.comparison.**Comparable**(\**args*, \*\**kwargs*)

Bases: object

Base class for conditions, overriding logical operators. Check if property exists by default.

```
EQ = 'eq'
EXISTS = 'exists'
GT = 'gt'
GTE = 'gte'
LIKE = 'like'
LT = 'lt'
LTE = 'lte'
NE = 'ne'
```

### link.dbrequest.assignment module

```
class link.dbrequest.assignment.A(propname, val=None, unset=False, *args, **kwargs)
```

Bases: `link.dbrequest.tree.Node`

Node representing an assignment.

#### Parameters

- **propname** (*str*) – Name of assigned property
- **val** (*Node*, or *Python type*) – Value to assign to property (ignored if unset is True)
- **unset** (*bool*) – Unset the property (default: False)

```
get_ast()
```

### link.dbrequest.expression module

```
class link.dbrequest.expression.CombinableExpression
```

Bases: `object`

Base class for expressions, overriding mathematical expressions.

```
ADD = 'add'
```

```
BITAND = 'and'
```

```
BITLSHIFT = 'lshift'
```

```
BITOR = 'or'
```

```
BITRSHIFT = 'rshift'
```

```
BITXOR = 'xor'
```

```
DIV = 'div'
```

```
MOD = 'mod'
```

```
MUL = 'mul'
```

```
POW = 'pow'
```

```
SUB = 'sub'
```

```
class link.dbrequest.expression.CombinedExpression(left, operator, right, *args, **kwargs)
```

Bases: `link.dbrequest.tree.Node`, `link.dbrequest.expression.CombinableExpression`

Combination of two expressions.

**Parameters**

- **left** (*CombinableExpression*) – left expression
- **operator** (*str*) – combination expression
- **right** (*CombinableExpression*) – right expression

**get\_ast** ()**class** `link.dbrequest.expression.E` (*name*, \**args*, \*\**kwargs*)Bases: `link.dbrequest.tree.Node`, `link.dbrequest.expression.CombinableExpression`

Expression referencing a property, usable in mathematical expressions to represent a formula applied on the referenced property.

**get\_ast** ()**class** `link.dbrequest.expression.F` (*funcname*, \**arguments*)Bases: `link.dbrequest.expression.E`

Function, representing a mathematical function.

**Parameters**

- **funcname** (*str*) – Mathematical function
- **arguments** (*list*) – List of expressions/values to use as function arguments

**get\_ast** ()**Module contents**



---

**Donating**

---



I

`link.dbrequest`, [15](#)  
`link.dbrequest.assignment`, [14](#)  
`link.dbrequest.ast`, [11](#)  
`link.dbrequest.comparison`, [13](#)  
`link.dbrequest.driver`, [7](#)  
`link.dbrequest.expression`, [14](#)  
`link.dbrequest.lazy`, [9](#)  
`link.dbrequest.manager`, [6](#)  
`link.dbrequest.model`, [9](#)  
`link.dbrequest.tree`, [13](#)





## A

A (class in link.dbrequest.assignment), 14  
 ADD (link.dbrequest.expression.CombinableExpression attribute), 14  
 all() (link.dbrequest.manager.QueryManager method), 6  
 AND (link.dbrequest.comparison.CombinableCondition attribute), 13  
 AST (class in link.dbrequest.ast), 11  
 ASTError, 11  
 ASTInvalidFormatError, 11  
 ASTInvalidStatementError, 12  
 ASTLastStatementError, 12  
 ASTSingleStatementError, 12

## B

BITAND (link.dbrequest.expression.CombinableExpression attribute), 14  
 BITLSHIFT (link.dbrequest.expression.CombinableExpression attribute), 14  
 BITOR (link.dbrequest.expression.CombinableExpression attribute), 14  
 BITRSHIFT (link.dbrequest.expression.CombinableExpression attribute), 14  
 BITXOR (link.dbrequest.expression.CombinableExpression attribute), 14

## C

C (class in link.dbrequest.comparison), 13  
 CombinableCondition (class link.dbrequest.comparison), 13  
 CombinableExpression (class link.dbrequest.expression), 14  
 CombinedCondition (class in link.dbrequest.comparison), 13  
 CombinedExpression (class link.dbrequest.expression), 14  
 Comparable (class in link.dbrequest.comparison), 13  
 copy() (link.dbrequest.lazy.Procedure method), 10  
 copy() (link.dbrequest.lazy.Query method), 10  
 count() (link.dbrequest.lazy.Query method), 10

count\_elements() (link.dbrequest.driver.Driver method), 8  
 create() (link.dbrequest.manager.QueryManager method), 6  
 Cursor (class in link.dbrequest.model), 9  
 cursor (link.dbrequest.model.Cursor attribute), 9  
 cursor\_class (link.dbrequest.driver.Driver attribute), 8

## D

data (link.dbrequest.model.Model attribute), 9  
 Dataset (class in link.dbrequest.lazy), 9  
 delete() (link.dbrequest.lazy.Query method), 10  
 delete() (link.dbrequest.model.Model method), 9  
 DIV (link.dbrequest.expression.CombinableExpression attribute), 14  
 Driver (class in link.dbrequest.driver), 7  
 driver (link.dbrequest.model.Cursor attribute), 9  
 driver (link.dbrequest.model.Model attribute), 9

## E

E (class in link.dbrequest.expression), 15  
 EQ (link.dbrequest.comparison.Comparable attribute), 13  
 exclude() (link.dbrequest.lazy.Query method), 10  
 execute() (link.dbrequest.lazy.Lazy method), 10  
 execute() (link.dbrequest.lazy.Procedure method), 10  
 execute() (link.dbrequest.lazy.Query method), 10  
 execute() (link.dbrequest.manager.QueryManager method), 6  
 in EXISTS (link.dbrequest.comparison.Comparable attribute), 14

## F

F (class in link.dbrequest.expression), 15  
 filter() (link.dbrequest.lazy.Query method), 10  
 find\_elements() (link.dbrequest.driver.Driver method), 8  
 find\_walker() (link.dbrequest.ast.NodeWalker method), 12  
 from\_ast() (link.dbrequest.manager.QueryManager method), 6

## G

`get()` (link.dbrequest.lazy.Query method), 11  
`get()` (link.dbrequest.manager.QueryManager method), 7  
`get_ast()` (link.dbrequest.assignment.A method), 14  
`get_ast()` (link.dbrequest.comparison.C method), 13  
`get_ast()` (link.dbrequest.comparison.CombinedCondition method), 13  
`get_ast()` (link.dbrequest.expression.CombinedExpression method), 15  
`get_ast()` (link.dbrequest.expression.E method), 15  
`get_ast()` (link.dbrequest.expression.F method), 15  
`get_ast()` (link.dbrequest.tree.Node method), 13  
`get_ast()` (link.dbrequest.tree.Value method), 13  
`group()` (link.dbrequest.lazy.Query method), 11  
GT (link.dbrequest.comparison.Comparable attribute), 14  
GTE (link.dbrequest.comparison.Comparable attribute), 14

## L

Lazy (class in link.dbrequest.lazy), 9  
LIKE (link.dbrequest.comparison.Comparable attribute), 14  
link.dbrequest (module), 15  
link.dbrequest.assignment (module), 14  
link.dbrequest.ast (module), 11  
link.dbrequest.comparison (module), 13  
link.dbrequest.driver (module), 7  
link.dbrequest.expression (module), 14  
link.dbrequest.lazy (module), 9  
link.dbrequest.manager (module), 6  
link.dbrequest.model (module), 9  
link.dbrequest.tree (module), 13  
LT (link.dbrequest.comparison.Comparable attribute), 14  
LTE (link.dbrequest.comparison.Comparable attribute), 14

## M

MOD (link.dbrequest.expression.CombinableExpression attribute), 14  
Model (class in link.dbrequest.model), 9  
model\_class (link.dbrequest.driver.Driver attribute), 8  
ModelBuilder (class in link.dbrequest.ast), 12  
MUL (link.dbrequest.expression.CombinableExpression attribute), 14

## N

name (link.dbrequest.ast.AST attribute), 11  
name (link.dbrequest.driver.Driver attribute), 8  
NE (link.dbrequest.comparison.Comparable attribute), 14  
Node (class in link.dbrequest.tree), 13  
NodeWalker (class in link.dbrequest.ast), 12

## O

OR (link.dbrequest.comparison.CombinableCondition attribute), 13

## P

`parse()` (link.dbrequest.ast.ModelBuilder method), 12  
POW (link.dbrequest.expression.CombinableExpression attribute), 14  
`prepare()` (link.dbrequest.manager.QueryManager method), 7  
Procedure (class in link.dbrequest.lazy), 10  
`process_query()` (link.dbrequest.driver.Driver method), 8  
`put_element()` (link.dbrequest.driver.Driver method), 8

## Q

Query (class in link.dbrequest.lazy), 10  
QUERY\_COUNT (link.dbrequest.driver.Driver attribute), 7  
QUERY\_CREATE (link.dbrequest.driver.Driver attribute), 7  
QUERY\_DELETE (link.dbrequest.driver.Driver attribute), 7  
QUERY\_READ (link.dbrequest.driver.Driver attribute), 7  
QUERY\_RUN (link.dbrequest.driver.Driver attribute), 7  
QUERY\_UPDATE (link.dbrequest.driver.Driver attribute), 8  
QueryManager (class in link.dbrequest.manager), 6

## R

`remove_elements()` (link.dbrequest.driver.Driver method), 8  
`run()` (link.dbrequest.manager.QueryManager method), 7  
`run_procedure()` (link.dbrequest.driver.Driver method), 8

## S

`save()` (link.dbrequest.model.Model method), 9  
SUB (link.dbrequest.expression.CombinableExpression attribute), 14  
`subset()` (link.dbrequest.manager.QueryManager method), 7

## T

`to_ast()` (link.dbrequest.lazy.Procedure method), 10  
`to_ast()` (link.dbrequest.lazy.Query method), 11  
`to_model()` (link.dbrequest.model.Cursor method), 9

## U

`update()` (link.dbrequest.lazy.Query method), 11  
`update_elements()` (link.dbrequest.driver.Driver method), 8

## V

val (link.dbrequest.ast.AST attribute), 11

`validate_ast()` (link.dbrequest.manager.QueryManager  
static method), [7](#)

`Value` (class in link.dbrequest.tree), [13](#)

## W

`walk()` (link.dbrequest.ast.NodeWalker method), [12](#)

## X

`XOR` (link.dbrequest.comparison.CombinableCondition  
attribute), [13](#)