MySQL Cellophane - Ruby Edition

Proof | Docs | Use | Codez

Getting Started

MySQL Cellophane has attributes for the basic builing blocks of a SQL query.

- table
  - specifies the table to operate on
- id_field
  - specifies which column in the main condition (used with id)
  - is set to 1 by default to override the main condition (id must be set to 1 as well)
- id
  - specifies a value for the main condition (used with field_id)
  - is set to 1 by default to override the main condition (id_field must be set to 1 as well)
  - is escaped before being placed into query
- extra_cond
  - Concatinated onto the end of the base query
  - IS NOT ESCAPED - use esc method to escape values before setting

MySQL Cellophane has methods for each basic SQL interaction:
Each interaction method can optionally take a final hash argument. This hash can set any of the class attributes listed above. See the examples below for more information.

- select
  - performs a select query
- insert({hash})
  - requires a hash argument of symbol keys for column names and values for each column to be inserted
- update
  - requires a hash argument of symbol keys for column names and values for each column to be updated
- delete
  - performs a delete query

Examples

friends table

id	name	age
1	Chris	26
2	Hayley	24
3	Bobby	25

initialize

```
db = MySQLc.new {:user => 'mysql_user', :password => 'mysql_password', :database => 'database'}
```

select

get all rows

```

```

```
db.table = 'friends'
        result = db.select # SELECT * FROM friends WHERE 1=1
        result.fetch_hash #=> { 'id' => 1, 'name' => 'Chris',  'age' => 26 }
        result.fetch_hash #=> { 'id' => 2, 'name' => 'Hayley', 'age' => 24 }
        result.fetch_hash #=> { 'id' => 3, 'name' => 'Bobby',  'age' => 25 }
```

get one friend

```
db.id_field = 'id'
        db.id       = 1
        result = db.select # SELECT * FROM friends WHERE id = 1
        result.fetch_hash #=> { 'id' => 1, 'name' => 'Chris',  'age' => 26 }
```

Notice how we don't need to specify the table again because we did in the example above. One of the very nice features of MySQL Cellophane is that it keeps your values for you.
setting the options in-line

```
result = db.select({ :table => 'friends', :id_field => 'id', :id => 2}) # SELECT * FROM friends WHERE id = 2
        result.fetch_row #=> [ 2, 'Hayley', 24 ]
```

specifying columns

```
db.field_list = 'name, age'
        db.id = 1
        db.id_field = 1
        result = db.select # SELECT name, age FROM friends WHERE 1=1
        result.fetch_row #=> [ 'Chris', 26 ]
        result.fetch_row #=> [ 'Hayley', 24 ]
```

using extra_cond

```
db = MySQLc.new
        db.table = 'friends'
        db.field_list = 'name'
        db.extra_cond = "name LIKE '%y' ORDER BY name"
        result = db.select # SELECT * FROM friends WHERE 1=1 AND name LIKE '%y' ORDER BY name
        result.fetch_row #=> [ 'Bobby' ]
        result.fetch_row #=> [ 'Hayley' ]
```

insert

```
db.insert { :name => 'Katie', :age => 23 }
```

new friends table

id	name	age
1	Chris	26
2	Hayley	24
3	Bobby	25
4	Katie	23

inline

```
new_id = db.insert { :name => 'Jared', :age => 26 }, { :table => 'friends' }
        new_id  #=> 5
```

new new friends table

id	name	age
1	Chris	26
2	Hayley	24
3	Bobby	25
4	Katie	23
5	Jared	26

update

```
db.id_field = 'name'
        db.id       = 'Katie'
        db.update { :age => 22 }
        db.select.fetch_hash #=> { 'id' => 4, 'name' => 'Katie',  'age' => 22 }
```

```
db.update { :age => 27 }, { :id => 26, :id_field => 'age', :table => 'friends' }
        db.id = 27
        db.select.fetch_hash #=> { 'id' => 1, 'name' => 'Chris',  'age' => 27 }
        db.select.fetch_hash #=> { 'id' => 5, 'name' => 'Jared',  'age' => 27 }
```

delete

```
db = MySQLc.new
        db.table = 'friends'
        db.extra_cond = 'age < 25'
        db.delete
```

id	name	age
1	Chris	26
3	Bobby	25
5	Jared	26