بسم الله الرحمن الرحیم

کتابخانه ارتباط و دریافت اطلاعات از پایگاه داده

**Mysql Query Library**

**نویسنده: رضا محیطی**

[**rm.biqarar@gmail.com**](mailto:rm.biqarar@gmail.com)

**09109610612**

**1392-12-23**

فهرست

[مقدمه 5](#_Toc388845338)

[آماده سازی کتابخانه 5](#_Toc388845339)

[توابع ورودی: 6](#_Toc388845340)

[آرگومان های توابع ورودی 6](#_Toc388845341)

[تابع field() 7](#_Toc388845342)

[String 7](#_Toc388845343)

[Array 7](#_Toc388845344)

[تابع where() 7](#_Toc388845345)

[String 7](#_Toc388845346)

[array 8](#_Toc388845347)

[تابع query 9](#_Toc388845348)

[توابع مکمل 9](#_Toc388845349)

[Limit() 9](#_Toc388845350)

[Order() 9](#_Toc388845351)

[Group() 10](#_Toc388845352)

[Join() 10](#_Toc388845353)

[string 10](#_Toc388845354)

[array 10](#_Toc388845355)

[توابع خروجی 11](#_Toc388845356)

[تابع string() 11](#_Toc388845357)

[تابع result() 11](#_Toc388845358)

[تابع error() 11](#_Toc388845359)

[تابع assoc() 12](#_Toc388845360)

[تنظیم دستی شماره رکورد یافت شده 13](#_Toc388845361)

[ذخیره سازی تغییرات بر روی رکورد های دریافت شده 13](#_Toc388845362)

[تابع object() 14](#_Toc388845363)

[تابع allAssoc() 14](#_Toc388845364)

[تابع allObject() 14](#_Toc388845365)

[تابع num() 14](#_Toc388845366)

[تابع LAST\_INSERT\_ID() 14](#_Toc388845367)

[توابع داخلی 14](#_Toc388845368)

[توجه! 14](#_Toc388845369)

[construct()\_\_ 15](#_Toc388845370)

[err() 15](#_Toc388845371)

[connection() 15](#_Toc388845372)

[connect() 15](#_Toc388845373)

[send() 16](#_Toc388845374)

[make\_string\_query() 16](#_Toc388845375)

[config() 16](#_Toc388845376)

[black\_field() 16](#_Toc388845377)

[splitor() 17](#_Toc388845378)

[update() 17](#_Toc388845379)

[select() 17](#_Toc388845380)

[delete() 17](#_Toc388845381)

[insert() 17](#_Toc388845382)

[table() 17](#_Toc388845383)

[save() 18](#_Toc388845384)

[onSave() 18](#_Toc388845385)

[endSave() 18](#_Toc388845386)

[\_return() 18](#_Toc388845387)

[record\_is\_called() 18](#_Toc388845388)

[check\_i() 19](#_Toc388845389)

# مقدمه

برای اتصال موتور php به موتور mysql و در یافت اطلاعات از پایگاه داده، مراحلی باید طی شود، و قوانین خاصی نیز باید رعایت شده تا به اطلاعات دست پیدا کنیم.

بدین منظور و در راستای سرعت بخشیدن به این مهم، کتابخانه ای طراحی شد.

این کتابخانه شامل سه قسمت اصلی است:

1. توابع ورودی(امکاناتی که کاربر برای آماده سازی هر نوع query در اختیار دارد)

2. توابع خروجی(اطلاعاتی که پس از ارتباط با پایگاه داده مورد نیاز کاربر است)

3. توابع داخلی (اقدامات لازم جهت آماده سازی query )

در این میان فقط توابع داخلی به صورت private و بقیه به صورت public هستند.

برای شروع کار کافی است این کتابخانه را در کد خود تزریق کنید

$sql = new sql\_lib;

و برای هر عملیاتی از توابع ورودی آن استفاده کنید

# آماده سازی کتابخانه

برای تنظیم کردن کتابخانه بر روی موتور MySql خود به دو روش می توانید عمل کنید

ابتدا اینکه وارد تابع connection() شوید و به ترتیب host ، username , password و نام دیتابیس را وارد کنید

$sql->connection("localhost","root","password","database\_name");

و یا اینکه متغیر های ایستا با نام های زیر در کد خود تعریف کنید.

define("db\_host", "localhost");

define("db\_user", "root");

define("db\_password", "");

define("db\_name", "database\_name ");

**لازم به ذکر است نام دیتابیس شما در تمامی query ها قبل از نام جدول وارد می شود.**

**Table\_name = database\_name + ‘.’ + table\_naem**

# توابع ورودی:

توابع ورودی عبارتند از:

**Select() insert() update() delete() query() transaction()**

**Commit() rollback() connection()**

این امکان وجود دارد که بارها و بارها در کد خود از این توابع استفاده کنید بدون اینکه مقادیر وارد شده در تابع قبل به فراخوان جدید تابع نفوذ کند. مثال:

**@example** $sql->select(table\_name\_1)

$sql->select(table\_name\_2)

**@return** select \* from table\_name\_1

Select \* from table\_name\_2

## آرگومان های توابع ورودی

این توابع اولین آرگومانی که دریافت می کنند، نام جدولی است که می خواهند بر روی آن عملیات انجام دهند.

البته می توان نام جدول را به صورت جدا نیز در این کتابخانه وارد کرد با تابع table() پس :

**@example** $sql->select(table\_name) **Or** $sql->select()->table(table\_name)

**@return** select \* from table\_name

و همین طور توابع ورودی دیگر. (نام جدول را می توان به صورت جدا به موتور ارسال کرد)

دومین آرگومانی که توابع ورودی می توانند دریافت کنند، بسته به نوع تابع ورودی متفاوت است که در جدل زیر به آن قید شده است. (البته تمامی این آرگومان ها را می توان به صورت جدا و در یک تابع مجزا وارد نمود)

|  |  |  |  |
| --- | --- | --- | --- |
| نام تابع | آرگومان اول | آرگومان دوم | آرگومان سوم |
| Select() | Table | Field | Where |
| Update() | Table | Field | Where |
| Insert() | Table | Field | Query |
| Delete() | Table | Where | limit |
| Query() | Query | - | - |
| roolback() | - | - | - |
| Commit() | - | - | - |
| Transaction() | - | - | - |

## تابع field()

این تابع نوع فیلد های مورد نیاز (چه برای نمایش و چه برای ویرایش و اضافه کردن) را مشخص می کند.

ورودی این تابع می تواند به دو روش باشد.

### String

زمانی که تابع select فراخوانی شده باشد می توان فیلد های مورد نیاز را به صورت string به این تابع ارسال کرد.

**@example** $sql->select(table\_name)->field(“id, name, date”)

**@return** select id, name, date from table\_name

اگر تابع field در حالت select فراخوانی نشود، سیستم به جای فیلد ها کاراگتر "\*" را قرار می دهد.

یعنی:

**@example** $sql->select(table\_name(

**@return** select **\*** from table\_name

### Array

اما زمانی که یکی از توابع update() و یا insert() فراخوانی شده باشند، از آنجا که نیاز است علاوه بر نام فیلد، مقدار فیلد هم در query وارد شود باید از آرایه استفاده کرد. مثال:

**@example** $sql->update(table\_name)->field(**array(“id” => 1, “name” => “ali”, “date” => 13930101))**

**@return** update table\_name SET **id = 1, name = ‘ali’ , date = 13930101**

در این زمان اندیس های آرایه به عنوان فیلد جدول ها به حساب می آیند و مقدار هر اندیس، مقدار فیلد ها.

## تابع where()

این تابع فقط در زمانی که یکی از سه تابع select , update , insert فراخوانی شده باشد، فعال است و می تواند شرط مورد نظر ما را در query اضافه کند.

ورودی این تابع می تواند به دو حالت باشد.

### String

دقیقا همان مقداری که به صورت string در این تابع وارد می شود در query بعد از کلمه WHERE وارد می شود

مثال:

**@example** $sql->select(table\_name)->where**(“id like (1) or name = ‘reza’”)**

**@return** select \* from table\_name WHERE **I’d like (1) or name = ‘reza’**

### array

اگر تعدادی شرط یکسان داشته باشید می تواند از آرایه استفاده کنید. به این صورت که اندیس آرایه نام فیلد، مقدار هر اندیس، مقدار شرط آن فیلد و بعد از تمام شدن آرایه در آرگومان دوم این تابع مشخص کنید که می خواهید بین این شرط ها از چه اپراتوری (= و LIKE و NULL و ... ) استفاده کنید و در آرگومان سوم این تابع جمله بین شرط ها مشخص می شود که یا “And” است و یا “Or”

به مثال های زیر توجه کنید:

**@example** $sql->select(table\_name)->where(array(“id” => 1 , “name” =>”ali”, “date” => 1393), “like” , “or”)

**@return** select \* from table\_name WHERE id like 1 or name like ‘ali’ or date like 1393

**@example** $sql->select(table\_name)->where(array(“id” => 1 , “name” =>”ali”, “date” => 1393), “<>” )

**@return** select \* from table\_name WHERE id <> 1 AND name <> ‘ali’ AND date <> 1393

**@example** $sql->select(table\_name)->where(array(“id” => 1 , “name” =>”ali”, “date” => 1393))

**@return** select \* from table\_name WHERE id = 1 AND name = ‘ali’ AND date = 1393

همان طور که در مثال ملاحضه کردید اگر آرگومان دوم به این تابع ارسال نشود به صورت خودکار کاراکتر "=" را جایگزاری می کند و اگر آرگومان سوم به این تابع ارسال نشود "AND" جایگزین می شود.

نکته: در PHP 5.4 به بعد میتوان از روش زیر برای آرایه ها استفاده کرد.

$sql->where([  
    'status' => 10,  
    'type' => 2,  
    'id' => [4, 8, 15, 16, 23, 42],  
]);

نتیجه

WHERE (`status` = 10) AND (`type` = 2) AND (`id` IN (4, 8, 15, 16, 23, 42))

اگر بخواهید از طریق id شرطی را ایجاد کنید علاوه بر اینکه از تابع where می توانید استفاده کنید راه آسان تری نیز پیش بینی شده است و آن تابع id است . یعنی:

**@example** $sql->select(table\_name)->id(100)

**@return** select \* from table\_name WHERE id = 100

در تمامی مواردی که شما نام فیلدی به دیتابیس ارسال می کنید اگر نام فیلد یکی از کد های موتور mysql باشد به صورت خودکار نام جدول قبل از آن قرار می گیرد.

بر فرض شما در یکی از جدول های خود فیلدی با نام “group” دارید و می خواهید به آن چیزی اضافه کنید و می دانیم که کلمه group یکی از کد های موتور mysql است و اگر به صورت تنها فراخوانی شود درخواست ما با خطا مواجه خواهد شد.

**@example** $sql->update(table\_name)->field(array(“id” => 1 , “name” =>”ali”, **“group” =>”نام گروه”))**

**@return** update database\_name.tabel\_name SET id = 1, name = ‘ali’ , **table\_name.group = ‘نام گروه’**

## تابع query

به دلیل متفاوت بودن برخی از query ها این امکان گذاشته شد که کاربر بتواند با توجه به نیاز خود query را ارسال کند

برای این کار از تابع query استفاده می کنید که دقیقا همان چیزی که به صورت string در آن وارد می کنید به موتور mysql ارسال می شود.

**@example** $sql->query(“select count(\*) from information\_shema.table where table name = ‘a’”)

**@return** select count(\*) from information\_shema.table where table name = ‘a’

توابع **transaction ، rollback** و **commit** نیز کاربر خاص خودشان را دارند که برای شما پوشیده نیست.

# توابع مکمل

## Limit()

این تابع دو عدد می گیرد. اولی شروع لیمیت دومی میزان ادامه دادن آن. شما زمانی می خواهید رکورد 1000 تا 1100 یک جدول را ملاحضه کنید به روش زیر می توانید عمل کنید.

**@example** $sql->select(table\_name)->limit(1000, 100)

اگر یک آرگومان به این تابع ارسال شود، آن عدد ارسال شده به عنوان عدد دوم لحاظ شده و عدد اول صفر می باشد یعنی:

**@example** $sql->select(table\_name)->limit(1100)

**@return** Select \* from table\_name limit 0 , 1100

## Order()

برای مرتب سازی query ها دریافتی از تابع order استفاده می کنیم به این صورت که آرگومان اول این تابع نام فیلدی است که می خواهیم اطلاعات آن را مرتب (sort) کنیم و آرگومان دوم آن اگر وارد **نشود** و یا **a-z** و یا **az** وارد شود آن فیلد را به صورت **صعودی** مرتب می کند اگر هم چیز دیگر **غیر از این ها** در آرگومان دوم وارد شود به صورت نزولی مرتب می کند.

**مرتب سازی صعودی**

**@example** $sql->select(table\_name)->order(“name”, “a-z”)

**@example** $sql->select(table\_name)->order(“name”, “az”)

**@example** $sql->select(table\_name)->order(“name”)

**مرتب سازی نزولی**

**@example** $sql->select(table\_name)-> order(“name”, “za”)

**@example** $sql->select(table\_name)-> order(“name”, 123)

## Group()

برای گروه بندی یک فیلد از این روش استفاده می شود، این تابع فقط نام فیلدی که می خواهید گروه بندی کنید را دریافت می کند.

در این قسمت نیز اگر فیلد فراخوانی شده یکی از کد های موتور mysql بود به صورت خودکار نام جدول قبل از فیلد قرار می گیرد.

**@example** $sql->select(table\_name)-> group(“gropu”)

**@return** select \* from table\_name group by table\_name.group

## Join()

در برخی موارد نیاز است که اطلاعات از چند جدول گرفته شود برای این کار از روش INNER JOIN استفاده می شود.

شما می توانید به میزان دلخواه از این تابع در یک درخواست از mysql استفاده کنید. و اطلاعات را از جدول های مختلف فراخوانی و دریافت نمایید.

به دو روش می توانید از این تابع استفاده کنید.

### string

اول اینکه شما به صورت دستی و String به این تابع مقادیر را ارسال کنید، که در این صورت تنها کاری که این تابع انجام می دهد این است که قبل از مقادیری که شماه وارد کرده اید جمله "INNER JOIN " را قرار می دهد.

**@example** $sql->select(table\_name)-> join(“table2”)

**@return** select \* from table\_name INNER JOIN table2

### array

اما روش دومی نیز در کار است، شما می توانید دو آرگومان به این تابع ارسال کنید.

اول نام جدولی است که می خواهید نام جدول اصلی query با آن Join شود. و دومین آرگومان، آرایه ای است متشکل از نام فیلد هایی که می خواهید با هم دیگر ارتباط داشته باشند

اندیس های این آرایه فیلد های جدول اصلی هستند و مقدار هر اندیس، فیلد های جدولی که می خواهید با آن ارتباط داشته باشد.

به مثال زیر توجه کنید.

**@ example** $sql->select("main\_table")->join("table\_1", array("id" => "id", "name" => "faname"))

**@return** SELECT \* FROM database\_name.main\_table INNER JOIN table\_1 ON main\_table.id=table\_1.id, main\_table.name=table\_1.faname

و یا در مثال زیر به بیش از یک بار از این تابع استفاده شده است:

**@example** $sql->select("main\_table")->join("table\_1", array("id" => "id", "name" => "faname"))-> join("table\_2", array("name" =>"name2"))

**@return** SELECT \* FROM database\_name.main\_table INNER JOIN table\_1 ON main\_table.id=table\_1.id, main\_table.name=table\_1.faname INNER JOIN table\_2 ON main\_table.name=table\_2.name2

# توابع خروجی

تمام مواردی که گفته شد، کتابخانه را برای برقراری ارتباط صحیح به موتور mysql آماده می کند. ولی خود ارتباط با موتور و دریافت query زمانی صورت می گیرد که شما یکی از توابع خروجی را فراخوانی کنید.

این کار از اتصال بی جهت دو موتور mysql و php جلوگیری می کند.

## تابع string()

اگر شما بخواهید متن خالص query که قرار است ارسال شود را ببینید می توانید از این تابع استفاده کنید.

این تابع ترجمه تمامی متغیر ها و تابع های فراخوانی شده را به شما نشان می دهد.

**تمامی مثال های بالا، با استفاده از این تابع نوشته شده بود.**

لازم به ذکر است این تابع با اینکه جزء توابع خروجی به حساب می آید ولی به موتور mysql متصل نمی شود.

## تابع result()

پاسخی که از موتور mysql دریافت می شود به صورت کامل و بدون هیچ گونه تغییری، خروجی این تابع است.

زمانی شما می خواهید خودتان پاسخ دیتابیس را بررسی و آنالیز کنید. این تابع برای چنین مواردی است و خروجی آن دقیقا خروجی کد زیر می باشد.

mysql\_query(string\_query)

## تابع error()

خطاهایی که به حسب نوع query ارسال شده ممکن است در mysql به وجود آید را می تواند با فراخوانی این تابع ببینید

با فراخوانی این تابع، query ساخته می شود و به موتور mysql ارسال می شود، سپس تنیجه کد mysql\_error() مورد بررسی قرار میگرد، اگر خطایی وجود نداشته باشد عبارت « خطایی وجود ندارد ☺ » را نمایش می دهد و اگر دارای خطا باشد، کد خطا و متن خطا را در قالب یک آرایه نمایش می دهد.

برای مثال شما در کوئری خود نام جدول را اشتباه وارد کرده اید و نام جدول در دیتابیس وجود ندارد در این حالت نتیجه این تابع عبارت است از:

array(

0 => 1146

1 => Table x doesn't exist

)

البته این امکان برای کل کلاس sql\_lib وجود دارد که به موتور خطای خارجی وصل شود. به این صورت که شما لینک موتور خطا را در هنگام فراخوانی کلاس sql\_lib به آن ارسال می کنید. در این حالت خطا ها علاوه بر تابع داخلی error در موتور خطای خارجی نیز وارد می شوند.

**$error = new error\_lib;**

**$sql = new sql\_lib($error);**

## تابع assoc()

تقریبا پر کاربر ترین تابع برای نمایش نتیجه دریافت شده از دیتابیس تابع assoc است .

این تابع و توابع همسو (allAssoc,object, allObject) در هنگام اولین فراخوانی، تمامی رکورد های دریافت شده از دیتابیس را در جایی ذخیره می کنند تا در فراخوانی های مجدد نیازی به دریافت اطلاعات از دیتابیس نباشد.

آرگومان هایی که این تابع دریافت می کند یا یک string است که نوع فیلدی را که می خواهید به شما نشان می دهد.

برای مثال

**@ example** $sql->select("main\_table")->allAssoc(“id”)

**@ return** 1,2,3,4,…….

نوع دیگری از آرگومان هایی که می تواند به این تابع ارسال شود function است.

گاهی شما می خواهید بر روی تمامی رکورهای دریافت شده از دیتابیس عملیاتی انجام دهید (مثلا تاریخ میلادی را به شمسی تبدیل کنید) کافیت یک از توابع assoc, allAssoc, object, allObject را فراخوانی کنید و در آرگومان این توابع، تابع خود را وارد کنید.

**@ example** $sql->select("main\_table")->allAssoc(function (){

# code ….

return $x;

});

البته تابعی که شما در آرگومان توابع خروجی می نویسید حتما باید return داشته باشند.

## تنظیم دستی شماره رکورد یافت شده

اگر بخواهید رکورد خاصی از لیست رکورد های یافت شده فراخوانی کنید کافیت عدد آن را در آرگومان یکی از دو تابع assoc , object وارد کنید

**@ example** $sql->select("main\_table")->assoc(10)

اگر آرگومان اول تابع اشغال بود می توانید این عدد را در آرگومان دوم وارد کنید.

**@ example** $sql->select("main\_table")->assoc(“id”, 10)

با این روش شما رکورد خاصی از لیست رکورد های دریاف شده را فراخوانی خواهید کرد.

توجه داشته باشید: فراخوانی ها از این به بعد به تعداد یک واحد به عددی که شما ارسال کرده اید اضافه می شوند

یعنی اگر شما به صورت دستی عدد 10 را در assoc وارد کنید و بعد از آن در خط بعدی مجدداassoc را فراخوانی کنید، رکورد 11 را به شما نمایش می دهد. و همین طور اگر در خط بعد تابع object را بخوانید، رکورد 12 را به شما نشان می هد.

ولی این تغییر اثری بر روی allAssoc , allObject ندارد و تمامی رکورد های دریافت شده را نمایش می دهد.

## ذخیره سازی تغییرات بر روی رکورد های دریافت شده

اگر شما بخواهید یک سری تغییرات بر روی رکورد ها انجام دهید و آنها را ذخیر کنید می توانید از روش زیر استفاده کنید.

ابتدا قبل از اینکه توابع allAssoc و یا allOjbect را فراخوانی کنید تابع save() را بخوانید.

این کار تمامی تغییراتی که با استفاده از تعریف function در دو تابع یاد شده اعمال می شود را ذخیره می کند و برای نمایش مقدار ذخیره شده کافیست تابع endSave() را فراخوانی کنید.

به مثال زیر توجه کنید:

**@ example** $sql->select("main\_table")->save()->allAssoc(function(){

$this->x = $x \* 1;

return $this;

})->endSave();

اگر بعد از ذخیره سازی یکی از رکورد ها را فراخوانی کنید تنیجه همان تنیجه ذخیره شده است.

## تابع object()

نوع دیگری از راه های دریافت اطلاعات از پایگاه، object است.

در این نوع (که تنها تفاوت میان این تابع و تابع assoc همین است) هر رکورد به صورت یک object نمایش داده می شود.

## تابع allAssoc()

این تابع تمامی رکورد های دریافت شده را به صورت یک آرایه نمایش می دهد.

## تابع allObject()

این تابع تمامی رکورد های دریافت شده را به صورت یک آرایه ولی با ازای هر اندیس یک object خروجی می دهد.

## تابع num()

این تابع تعداد رکورد های یافت شده را نمایش می دهد.

## تابع LAST\_INSERT\_ID()

آخرین id اضافه شده به دیتابیس را می تواند از طریق این تابع دریافت کرد. (کاربرد در هنگام فراخوانی تابع insert )

# توابع داخلی

## توجه!

توابع داخلی در دو کلاس sql\_lib و sql\_query قرار دارند.

وظیفه اصلی ساخت و آنالیز کوئری بر عهده کلاس sql\_query است و کلاس sql\_lib به استفاده همزمان از چند تابع ورودی مانند select, insert , … می باشد.

در کلاس sql\_lib تمامی توابع ورودی که در کلاس sql\_query قرار دارند تکرار شده اند با این تفاوت که در هر بار فراخوانی یکی از توابع sql\_lib از کل کلاس sql\_query یک new ساخته می شود تا اطلاعات توابع مکمل مانند where,field,join,… با یکدیگر ادغام نشوند.

با این کار شما می توانید در یک بار فراخوانی از تمامی توابع ورودی با متغیر های متفاوت استفاده کنید بدون آنکه به متغیر قبلی آسیب برسد.

برای مثال:

**@ example** $sql->select("table\_name ")->field(“id,name”)->where(array(“id”=>1, ”name”=>”ali”)

$sql->update("table\_name ")->field(array(“id”=>1, ”name”=>”ali”)->where(“id = 10”)

$sql->insert("table\_name ")->field(array(“id”=>1,”name”=>”ali”)

تمامی این کوئری ها از توابع مکمل field و یا where استفاده کرده اند ولی به صورت مجزا به موتور mysql ارسال می شوند و نتایج تفکیک شده ای به ما می دهند.

در اینجا به توضیح توابع داخلی کلاس sql\_query می پردازیم:

## construct()\_\_

این تابع تمامی اطلاعات موتور تنظیم خطای خارجی را (اگر در هنگام فراخوانی این کلاس، اطلاعات موتور تولید خطای خارجی به آن وارد شده باشد) در $this->error ذخیره می کند تا زمانی که در اجرای فرمان، خطایی به وجود آمد، علاوه بر تنظیم خطا در تابع error() همین کلاس ، در موتور خطای خارجی نیز تغییرات اعمال شود.

## err()

این تابع خطای به وجود آمده را دریافت می کند و در دو قسمت ذخیره می نماید، اول تابع خطای داخلی (همین کلاس)، دوم موتور خطای خارجی

## connection()

تنظیم متغیر های ارتباط با دیتا بیس، بر عهده این تابع است. این متغیر ها از نوع static هستند و در صورتی که در این تابع وارد نشوند به صورت خودکار از متغیر های ایستای db\_host,db\_user,db\_password,db\_naem اطلاعات خود را دریافت می کنند.

## connect()

اتصال دو موتور mysql و php در این تابع صورت می پذیرد، علاوه بر این کار، فقل کردن کلید اتصال برای جلوگیری از اتصال مجدد و ذخیره سازی اتصال جاری در این تابع است.

## send()

ارسال کوئری های تولید شده به موتور mysql از طریق این تابع است.

این تابع به دو حالت به موتور mysql کوئری را ارسال می کند. اول زمانی که آرگومانی به این تابع ارسال شود، که این در زمانی است که یکی از توابع ورودی commit() ، rollback() و یا trancaction() فراخوانی شوند. این توابع دستورات از پیش آماده شده ای را به طور مستقیم به موتور mysql ارسال می کنند ( که در آرگومان تابع send() آن دستورات وارد می شوند)

اگر هیچ آرگومانی به این تابع ارسال نشود، متن کوئری با استفاده از متغیر ها و توابع فراخوانی شده تولید می شود و پس از بررسی خطا به موتور mysql ارسال می شود.

در این میان اگر خطایی وجود داشته باشد، سیستم کلید ارسال را قفل می کند (تا از ارسال مجدد کوئری همراه با خطا جلو گیری شود)

زمانی که کوئری ارسال می شود ، خالص نتیجه ارسال ذخیره می شود تا در تمامی توابع به آن دسترسی داشته باشیم.

زمانی که سیستم در یکی از مراحل فوق دچار اشکال شود، بعد از تولید خطا false را خروجی می دهد.

## make\_string\_query()

تمامی توابع فراخوانی شده، متغیری را در $this ذخیره می کنند. وظیفه این تابع متصل کردن و در کنار هم قرار دادن مقادیر مورد نیاز با توجه به مدل اصلی کویری (insert,update,delete,select) است که نتیجه آن تولید string کوئری جهت ارسال به موتورmysql است.

در این تابع، نام دیتا بیس، همراه با کاراکتر «.» قبل از نام جدول جهت استاندارد سازی کوئری قرار می گیرد.

## config()

آرگومان هایی که به توابع ورودی select(),insert(),update(),delete() ارسال می شود، ابتدا وارد این تابع شده و مشخص می شود که هر آرگومان مربوط به کدام تابع داخلی است.

برای مثال شما در تابع select() سه آرگومان ارسال می کنید، در این تابع مشخص می شود که آرگومان اول نام جدول است، آرگومان دوم فیلد های مورد نظر شما و آرگومان سوم شرط می باشد و هر کدام از آرگومان ها را به توابع تخصصی خودشان ارجاع می دهد.

## black\_field()

کلماتی وجود دارند که اگر از آنها به عنوان نام فیلد های جدولتان استفاده کنید در هنگام ارسال کوئری دچار مشکل می شوید چون یکی از کد های موتورmysql می باشند، مانند group

در این تابع لیستی از آیتم ها وجود دارد که در زمانی که شما فیلدی را از دیتا بیس فراخوانی می کنید (جهت insert یا update) اگر نام فیلد یکی از آیتم های این تابع باشد، به صورت خودکار، نام جدول به همراه کاراکتر "." قبل از نام فیلد قرار می گیرد تا در موتور Mysql خطا رخ ندهد.

## splitor()

مقادیر ارسال شده در توابع field(),join(),where() که همگی متشکل از یک آرایه می باشند که اندیس آرایه ، نام فیلد و مقدار هر اندیس ، مقدار مورد نظر ما برای آن فیلد می باشند به این تابع ارسال می شوند ، تا در یک مکان واحد به آنالیز آنها بپردازیم.

اگر میزان ارسال شده string بود، همان مقدار را پاسخ دهد (بدون هیچ گونه دخل و تصرف) اگر array بود آن را آنالیز کند و field ها و مقدار هر فیلد مشخص شود.

## update()

تابع ورودی جهت تنظیم مدل اصلی کلاس برای تغییر در فیلد ها.

وظیفه این تابع تنها تنظیم کردن مدل اصلی کوئری بر update است تا در توابع دیگر استفاده شود.

## select()

تابع ورودی جهت تنظیم مدل اصلی کلاس برای دریافت اطلاعات.

وظیفه این تابع تنها تنظیم کردن مدل اصلی کوئری بر select است تا در توابع دیگر استفاده شود.

## delete()

تابع ورودی جهت تنظیم مدل اصلی کلاس برای حذف یک یا چند ردیف.

وظیفه این تابع تنها تنظیم کردن مدل اصلی کوئری بر delete است تا در توابع دیگر استفاده شود.

## insert()

تابع ورودی جهت تنظیم مدل اصلی کلاس برای اضافه کردن اطلاعات به جدول ها

وظیفه این تابع تنها تنظیم کردن مدل اصلی کوئری بر insert است تا در توابع دیگر استفاده شود.

## table()

نام جدولی که در حال انجام عملیات بر روی آن هستیم توسط فراخوانی این تابع و ارسال نام جدول به آن، تنظیم می شود.

زمانی که شما در یکی از توابع ورودی (select(),update(),insert(),delete()) نام جدول را وارد می کنید توسط تابع config نام جدول به این تابع ارسال می شود.

## save()

فعال سازی کلید ذخیره کوئری ها توسط این تابع انجام می شود.

زمانی که شما می خواهید بر روی کوئری های دریافت شده تغییرات ایجاد کنید و این تغییرات ذخیره شوند، ابتدا این تابع را فراخوانی می کنید، از این به بعد تمامی تغییرات بر روی رکورد های یافت شده ذخیره می شوند.

قابل استفاده در زمانی که در تابع خروجی allAssoc ، function تعریف می کنید و رکورد ها را تغییر می دهید.

## onSave()

در زمانی که کلید ذخیره سازی تغییرات بر روی رکورد ها (تابع save() ) فعال باشد، تمامی رکورد ها هنگام آنالیز و فراخوانی به این تابع ارسال می شوند تا در جایگاه خود ذخیره شوند.

## endSave()

نمایش تمامی رکورد ها و تغییرات اعمال شده بر روی آنها و نیز غیر فعال سازی کلید ذخیره سازی رکورد توسط این تابع انجام می شود.

## \_return()

توابع خروجی (assoc(),allAssoc(),object(),allObject()) مقادیر ارسال شده به خود را به این تابع می فرستند تا مورد بررسی قرار گیرد.

اگر یکی از مقادیر ارسال شده عدد بود، یعنی کاربر شماره رکورد خاصی را در نظر دارد و باید تنها همان رکورد به او نمایش داده شود.

اگر از function استفاده کرده بود، باید مقادیر به همان تابع کابر ارسال شود تا بتواند آنها را تغییر دهد

و یا اگر نام فیلد خاصی را مد نظر داشت ، فقط مقدار همان فیلد را نمایش می دهد.

و اگر هیچ آرگومانی به توابع یاد شده ارسال نشده بود، - طبق روال و آخرین رکورد نمایش داده شده - رکورد بعد را نمایش می دهد.

## record\_is\_called()

در این کلاس، تنها یک بار تمامی رکورد های یافت شده به صورت کامل توسط تابع mysqli\_fetch\_assoc تولید و ذخیره می شوند، در هر جایی اگر نیاز به رکورد های نتیجه شد، از رکورد های ذخیره شده استفاده می شود.

این کار از دسترس خارج شدن رکورد ها را از بین می برد و باعث می شود بتوان بارها و بارها تمامی رکورد ها را فراخوانی کرد.

علاوه بر این کلید دریافت رکورد ها را غیر فعال می کند تا در مراحل بعد مجددا mysqli\_fetch\_assoc اجرا نشود.

## check\_i()

شماره سراسری رکورد نمایش داده شده در این تابع ثبت و ذخیره می شود.

زمانی که شما، شماره ای برای رکورد خاصی در نظر دارید، و آن را به یکی از توابع (assoc(),allAssoc(),object(),allObject()) ارسال می کنید ، در حقیقت ، شماره سراسر تغییر کرده و در فراخوانی مجدد یکی از توابع یاد شده ، به این شماره سراسری یک واحد اضافه می شود.

exit();