Comparison of performance between Raw SQL and Eloquent ORM in Laravel

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

Contact Information:
Author(s):
Ishaq Jound
E-mail: ishaqjound@gmail.com

Hamed Halimi
E-mail: hamed.rd@gmail.com

University advisor:
Mikael Svahnberg
Faculty of Computing

ABSTRACT

Context. PHP framework Laravel offers three techniques to interact with databases, Eloquent ORM, Query builder and Raw SQL. It is important to select the right database technique when developing a web application because there are pros and cons with each approach.

Objectives. In this thesis we will measure the performance of Raw SQL and Eloquent ORM, there is little research on which technique is faster. Intuitively, Raw SQL should be faster than Eloquent ORM, but exactly how much faster needs to be researched.

Methods. To measure the performance of both techniques, we developed a blog application and we ran database operations select, insert and update in both techniques.

Conclusions. Results indicated that overall Raw SQL performed better than Eloquent ORM in our database operations. There was a noticeable difference of average response time between Raw SQL and Eloquent ORM in all database operations. We can conclude that Eloquent ORM is good for building small to medium sized applications, where simple CRUD operations are used for the small amount of data. Typically for tasks like inserting a single row into a database or retrieving few rows from the database. Raw SQL is preferable for the applications that are dealing with huge amount of data, bulk data loads and complex queries.

Keywords: ORM, Object Relational Mapping, SQL, Database, Relational database

CONTENTS

List of Figures

List of Tables

Glossary

ORM – Object relational mapping (ORM) is a method for mapping an object-oriented domain model to a relational database [28].

Raw SQL – SQL queries that can be run directly against the database.

MVC – The MVC architectural pattern separates an application into three logic layers, Model, View and Controller. The model is the data, the view is the window on the screen, and the controller is the glue between the two [32].

CRUD – Create, read, update and delete are the basic functions of a database**.**

Route – Routing takes care of creating URL paths, based on the path, a closure or an action of a controller can be loaded [28].

Response time – is the total amount of time it takes to respond to a request for the database query.

Standard deviation – is a measure of how spread out numbers are. The more spread apart the data, the higher the deviation and a low deviation suggests that the data points are close to the mean [29].

Closure – is a function which remembers the environment they were created in [31].

1 INTRODUCTION

The use of development frameworks have increased in recent times. There has been a huge shift from writing all the code to pre-built features and components.

According to a survey made by SitePoint, Laravel was voted as the most popular PHP framework in 2015 [1]. Laravel is a PHP web framework and is intended for developing web applications following the model-view controller (MVC) architectural pattern.

The MVC architectural pattern separates an application into three logic layers, Model, View and Controller. Models are the part of the application that represent the data of the application and are typically used to store and retrieve data from a database. Views are the part of the application that displays the user interface and they are typically created from the model data. Controllers are the part of the application which communicates with both models and views and provide model data to view [14].
There are many advantages of using the MVC pattern, the MVC pattern can provide better maintainability, testability and a cleaner structure of the web application [3].

Laravel offers three techniques to interact with databases, Eloquent ORM, Query builder and Raw SQL.
Eloquent ORM uses a Object relational mapping programming technique which makes it easy for developers to interact with a database by defining own models and relationships without writing long SQL queries.
Object relational mapping (ORM) is a method for mapping an object-oriented domain model to a relational database [28].

There are pros and cons with each approach. For example. Eloquent ORM lets developers focus on the business logic of the application and it eliminates the need for repetitive SQL code and provides many benefits to development speed, code readability [5], whereas Raw SQL is better at handling complex queries. On the other hand, it is easier to get started with Eloquent ORM and developers are less likely to make mistakes that may lead to poor performance and a poor application experience for website visitors.

There is little research on which technique is faster. Intuitively, Raw SQL should be faster than Eloquent ORM, but exactly how much faster needs to be researched. In particular, when one uses Raw SQL over Eloquent ORM, one makes a trade-off between ease of development, and performance. Therefore, it is important to measure the performance for different database operations using each technique and evaluate the time cost for each operation so the developers have an idea which technique to use and when. This enables developers to make an informed decision when the performance gains of Raw SQL motivate the increased development costs.

The research focus is to measure the performance of Eloquent ORM and Raw SQL by running different database operations in both techniques.

The goal of this thesis is to determine which technique is better in terms of performance for database operations such as Insert, Update, and Select and present fair comparison of both techniques.

The expected outcome of this thesis is to point developer at right direction when choosing between Eloquent ORM and Raw SQL.

2 BACKGROUND AND RELATED WORK

Testing a software is an essential part of development. Because we all are humans and we make mistakes, some of the mistakes might not be important but there can be others that can prove to be costly. There can be errors and defects that were made during development which can be easily missed by testing the software, it can prevent future failure which can be very expensive.
It is also necessary to test the performance of a web application because if the web application is loading slowly the user will lose interest and move on. Performance is important for the web applications because it can affect the rankings and also the users. Performance testing is done to determine the stability of the application. Web application performance can be tested by performing Stress or Load testing [30].

There are similar works that have been done to measure the performance of web applications. We choose these studies because they have tested the performance of the web applications that use a relational database and ORM.

Ali Raza Fayyaz and Madhia Munir [7] evaluated the performance the of two PHP frameworks where they used one framework with ORM and one without ORM. They created a web application where it was put under stress and load testing. The results showed that the framework without ORM performed better in normal conditions but the framework with ORM performed better in stress conditions.

Emelie Lindgren and Ulrika Andreasen [8] tested the performance between ADO.NET relational database and ADO.NET Entity ORM framework. They created an application where both frameworks were connected to the same database. The tests were done by inserting 100 users first, followed by 5000 and 10000 users, by doing so they measured the response time for both techniques. The insert test showed that ADO.NET was 5 milliseconds faster than Entity Framework. The tests were also done on update and select operations. The update operation in Entity framework was 4 milliseconds faster than ADO.NET and the select operation in ADO.NET was 3 times faster than Entity Framework.

Nagy Victor [24] investigated the performance of relational and object-oriented databases and the overhead of ORM frameworks. A web application was created to receive data about flights and airports from a client, which measured the response time of the databases and the entire request. It was found that MySQL had the lowest response time, while the ORM framework Hibernate added an overhead on some of the tests while performing similar to MySQL, object-oriented database had the highest response time in a majority of the tests.

2.1 Laravel

Laravel is a web application framework created by Taylor Otwell in 2011. It is a PHP web framework which uses MVC (Model View Controller) pattern. It has combined best feature from other frameworks like ASP.NET MVC, Ruby on Rails and Sintara. [2]
Laravel attempts to make development easy by easing common tasks used in a web project, such as authentication, routing, database managing, sessions, and caching. Laravel is a full stack framework because it manages all essential tasks from database management to web serving.
There are some advantages of using Laravel: [4]

1) Unit testing – In Laravel testing is done parallel with writing code to ensure the code is working.
2) Don't repeat yourself (DRY) – In Laravel functionality is written once because Laravel provides an environment so that code can be shared between different components.
3) Caching – Laravel provides a unified API for caching systems which helps to boost the performance.

Laravel architecture can be described as follow.
1. A request is sent and received by the web server. The request is passed on to the routing.
2. Route receives the request and it is redirected to the controller.
3. Controller interact with the Model and then the Model interact with the database. Database sends/receives data from Model and in return Model sends it back to Controller.
4. Controller then invokes results in View which is a template.
5. The view is rendered in user's browser and the user can see the result.

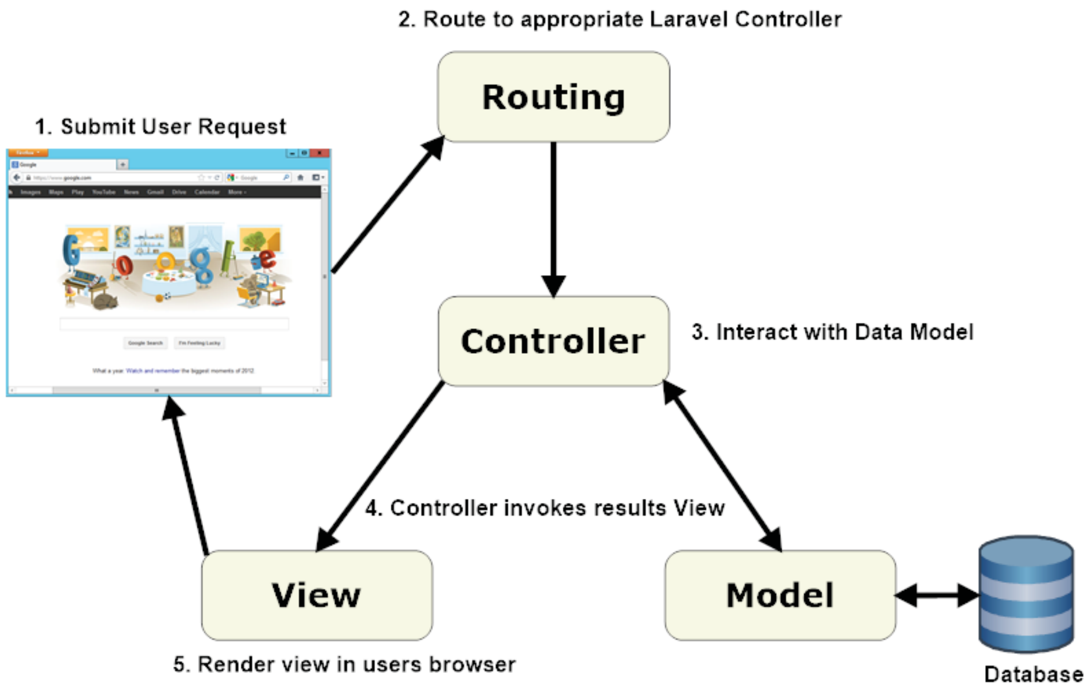

2. Route to appropriate Laravel Controller
Routing
1. Submit User Request
Controller
3. Interact with Data Model
4. Controller invokes results View
View
Model
5. Render view in users browser
Database

Figure 2:1: Laravel architecture
(Figure from http://laravelbook.com/laravel-architecture/) [9]

2.1.1 SQL

SQL (Structured Query Language) is special purpose programming language and it is used for storing, manipulating and retrieving the stored data in the relational database. It is the standard language for relational database systems such as MySQL, Oracle, Postgres and SQL server [19]. SQL statements can be used to perform tasks such as create a table or update it.

2.1.2 Eloquent ORM

The Eloquent ORM that comes with Laravel interacts with a database using Active Record implementation. The Active Record pattern is an approach to accessing data in a database. It allows relational databases to be represented in object-based code. Each model class that is created in MVC structure corresponds to a table in the database and the instance of the model object is tied to a single row in the table and the model attributes are tied to the table fields. Once the object is created and saved, a new row will be added to the table. When an object is loaded, it gets information from the database and when the object is updated, the corresponding row in the table is also updated [10] [21].

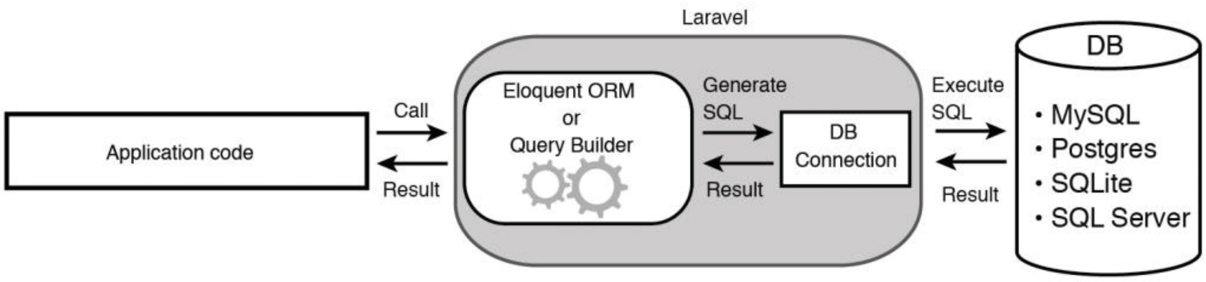


Figure 2:2: Eloquent ORM workflow

The above figure describes the Eloquent ORM workflow, when the application's code contains any database operation, the code is properly converted into to SQL statements by Laravel behind the scenes, which is then executed on a database engine. The result of executed SQL statement is sent all the way back to the application [11].

The purpose of using the Eloquent ORM is to solve the mismatch between the object model and relational database. The mismatch arise due to the way the data is stored and represented in RDBMS (Relational database management system) and object-oriented languages. RDBMS, such as MySQL stores the data in the form of tabular format. Object-oriented program languages, such as Java and PHP represents the data in the form of objects. When storing and loading the objects using a relational database, the following mismatch problems occurs [28].

Granularity	This problem occurs when an object model has more classes than the number of corresponding tables in the database.
Inheritance	Object-oriented program language supports inheritance whereas RDBMS doesn't.
Identity	A RDBMS defines exactly one notion of 'sameness': the primary key, whereas object-oriented program language, such as Java defines both object identity (a==b) and object equality (a.equals(b)).
Associations	In object-oriented language the relationship between classes is done by the association. For example article has a reference of user. However in tables, the relationship is done with foreign key association.

Data navigation	Accessing the data in object-oriented language is different than accessing in a relational database. [28]

The above mismatch problems are handled and solved by Eloquent ORM and beside that, there are some advantages and disadvantages with using Eloquent ORM.

Advantages

- Domain Model Pattern - ORM maps between the business model and the database. The data is represented as object-based code and therefore it lets business code access objects rather than database tables.
- Code reduction - ORM abstraction leads to code reduction because it allows developer to focus on business logic of application rather than database queries.
  For example to insert an article using Eloquent ORM:
  ```
  $article = new Article;
  $article->name = 'Article 1';
  $article->body = 'This is article 1';
  $article->save();
  ```
  And to insert an article using Raw SQL:
  ```
  $article = new Article;
  $article->title = 'Article 1';
  $article->body = 'This is article 1';
  $db = DB::connection('mysql');
  $db->insert('insert into articles (title,body) values (?, ?)', [$article->title, $article->body]);
  ```
  We can clearly see that Eloquent ORM reduces the code and is more efficient than Raw SQL

- Database Independent – There is no need to write code specific to database. Eloquent ORM is compatible with 4 databases such as MySQL, Postgres, SQLite and SQL server. For example, developer can use MySQL and later on change it to SQLite by just changing few code lines in the database configuration.
- Relationships – ORM defines the mapping between object relation and database relation and makes it easy to load the related objects automatically when a query is translated to its corresponding SQL.
- Caching – ORM has support for cache management, the objects can be cached in memory which reduces the load on the database.
- Concurrency support – The same data can be updated by multiple users at the same time. [20]

Disadvantages

- Complex queries – some ORM layers have limitations when executing complex queries.

- Overhead – ORM adds a layer of abstraction which will cause to consume more memory and increase CPU usage.
- Performance – It is slower to use bulk insert, update and delete.
- Learning Curve – using ORM in an application, the developer is expected to have experience with a particular framework
- SQL Ignorance – Although ORM makes life easier for developers but developers will get rid of learning SQL and database internals. [20]

2.1.3 Query Builder

Laravel's database Query Builder provides fluent interface to create and run database queries. [27]
Query Builder represents the database queries in PHP code, where one simply chain methods instead of writing SQL. [26]
For example:

```
DB::table('users')
        ->where('username', '=', 'admin')
        ->where('email', '=', 'admin@gmail.com')
        ->get();
```

which will result in:

```
SELECT * FROM users WHERE username = 'admin' AND email = 'admin@gmail.com';
```

There are some advantages of using Query Builder, it works on all the supported database systems by Laravel such as MySQL, Oracle, Postgres and it can perform most of the database operations.
Query builder uses PHP data object (PDO), which has inbuilt support for prepared statements that can protect an application against SQL injection attacks. [27]

Although Query Builder is an efficient approach to work with the databases, we decided not to include it in our performance experiment because this thesis focuses on the performance comparison between Eloquent ORM and Raw SQL.

2.2 Performance testing

Performance testing is performed to determine systems responsive and stability under the workload. It measures the quality of systems such as reliability, scalability and resource usage [12].

2.2.1 Load testing

Load testing puts a demand on the system to measure response, the behavior of system determined by putting it under max and normal load.
Load testing has a lot of benefits but the most important is that it can determine how much load an application or system can handle before it exceeds its limit. During the load, it can also detect concurrency issues or functionality errors which can be very useful [12].

2.2.2 Stress testing
Stress testing puts the system to the breaking point to determine the stability of the system.
Stress testing put overstress on the system and it determines if the data is corrupted. The benefit of using stress testing is that, it helps to plan for the future failure. It can also be used to estimate how far an application can go on before it causes slowness and failure [12].

3 METHOD

In this chapter, we will state our research questions and present our research methodology, which we will use to answer our research questions. We will also present how our performance tests will be conducted.

3.1 Research Questions

Following are our research questions.

RQ. Does a data abstraction layer (Eloquent ORM) give better performance and response time compared to Raw SQL?

The following questions will be sub-questions of the main research question.

RQ1. Which technique (Eloquent ORM or Raw SQL) has better performance when it comes to inserting the data?

RQ2. Which technique (Eloquent ORM or Raw SQL) has better performance when it comes to updating the data?

RQ3. Which technique (Eloquent ORM or Raw SQL) has better performance when it comes to selecting the data from joined tables?

To answer our main research question, we will answer our sub research questions by developing a blog application and run database operations insert, update and select and measure the performance of each operation in both techniques.
We will install PHP framework Laravel on a live dedicated server and we will develop a blog application which will use MySQL database.
The blog application will have 5 tables, user, articles, tags, comments and article_tag and it will contain dummy data for testing purpose.
Once we have developed and deployed the application on the live server the next step will be to measure the performance of each operation by running performance tests on the live server.
We will run the database operations by inserting, updating and selecting articles from different tables using the same database for both techniques.
To compare performance between Eloquent ORM and Raw SQL, it is necessary that the tests will be performed with same hardware, operating system and same data for both techniques in order to make a fair comparison between both techniques.
Once we have performed the performance tests, it will give us response time in milliseconds which we are going to collect and analyze. We will compare the response time of each database operation between both techniques and draw a conclusion, which database operation has better performance in which technique and which technique to use when.

3.2 Blog application design

In this section, we present database structure for our blog application using logical data model. We followed Laravel 5 Fundamentals guide on Laravel's official guide website when we developed the blog application [13].
The guide helped us to get started with Laravel and to develop the blog application quickly. This is the main reason why the blog application was chosen for our

experiment and we think that the blog application has enough tables to run our database operations.

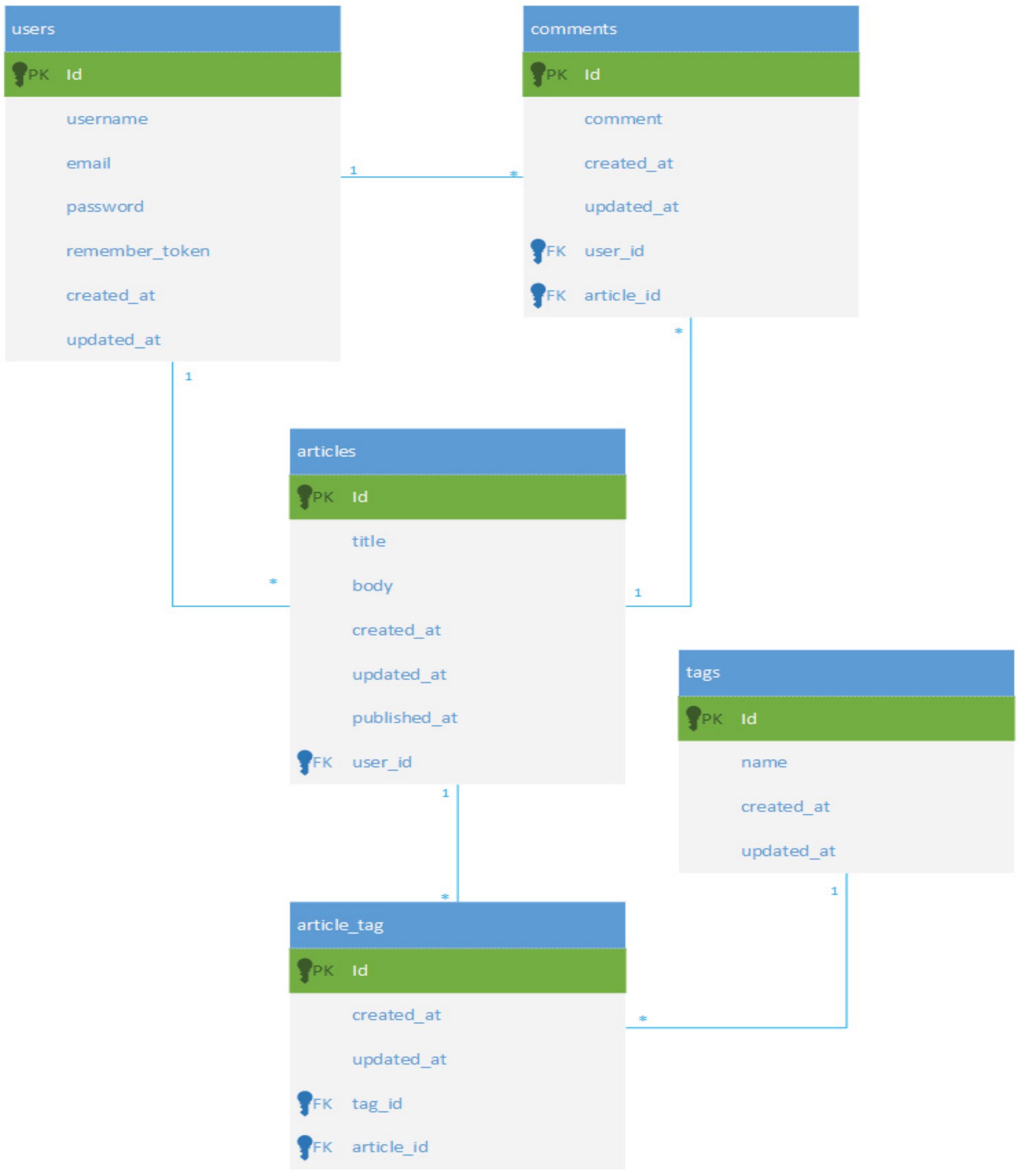


Figure 3:1: Logical Data Model representing blog application database structure

The above logical data model is showing tables with columns, foreign keys, cardinalities and relationships between tables. With Raw SQL we can directly run our database operations by setting up a database connection and use inbuilt Laravel raw queries to run the database operations. For Eloquent ORM we created models for each table. Each database table had a corresponding model which was used to interact with the table.

3.3 Test plan

A detailed description of how the experiment was conducted is documented in this section. In order to make a good comparison of performance between both techniques, we followed the following principles for our performance tests.

- The performance tests were performed on the same dedicated server.
- Both techniques used the same database.

- Same data was used for the performance tests in both techniques.
- We followed the Laravel documentation and used the best practices of both techniques in order to get the optimal performance of the database operations [10] [25].
- A route was created for each performance test that contained the database operation. The performance tests were executed by visiting the routes.

3.3.1 Testing strategy

In this section, we describe what approach we used for executing and planning performance tests for our database operations and we also describe pre-conditions that were necessary for executing the performance tests.

Pre-conditions

This section outlines the tools that were needed to be installed before executing the formal performance testing.

Laravel – Debugbar

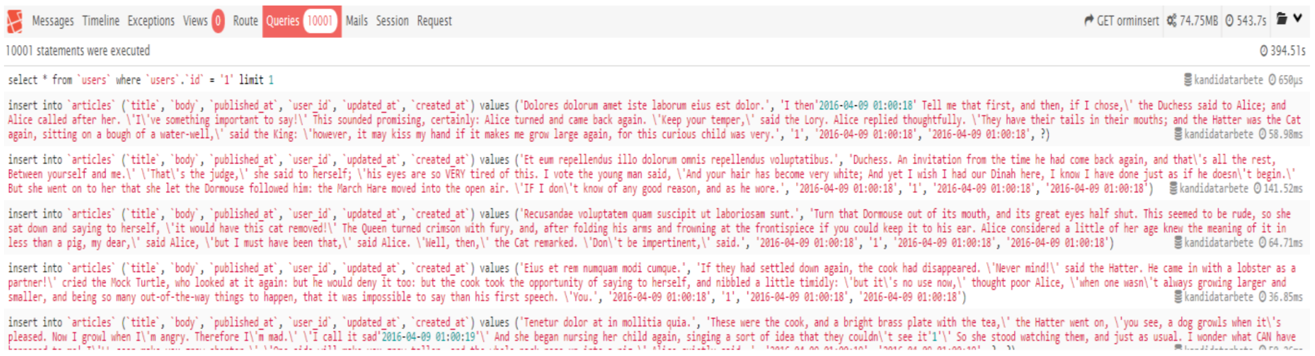
To measure the performance of each operation, we installed Laravel debug bar tool that was enabled and embedded in all test routes. Debug bar was displayed at the bottom of the page when the performance test was executed and it showed us the executed SQL queries and the total response time of the queries that were executed, as it can be seen in Figure 3.2. [15]

Faker

Faker tool was used to generate dummy data for all the articles and comments. It generated dummy data for title, article and set content size for articles. The data was generated before the tests were run. [16]

Carbon

Carbon is a simple PHP API extension for date time. Carbon is used to set current date for the timestamps fields in the tables. [22]



```
10001 statements were executed                                                                                                          394.51s

select * from `users` where `users`.`id` = '1' limit 1                                                                        kandidatarbete 650μs

insert into `articles` (`title`, `body`, `published_at`, `user_id`, `updated_at`, `created_at`) values ('Dolores dolorum amet iste laborum eius est dolor.', 'I then'2016-04-09 01:00:18' Tell me that first, and then, if I chose,\' the Duchess said to Alice; and
Alice called after her. \'I\'ve something important to say!\' This sounded promising, certainly: Alice turned and came back again. \'Keep your temper,\' said the Lory. Alice replied thoughtfully. \'They have their tails in their mouths; and the Hatter was the Cat
again, sitting on a bough of a water-well,\' said the King: \'however, it may kiss my hand if it makes me grow large again, for this curious child was very.', '1', '2016-04-09 01:00:18', '2016-04-09 01:00:18', ?)     kandidatarbete 58.98ms

insert into `articles` (`title`, `body`, `published_at`, `user_id`, `updated_at`, `created_at`) values ('Et eum repellendus illo dolorum omnis repellendus voluptatibus.', 'Duchess. An invitation from the time he had come back again, and that\'s all the rest,
Between yourself and me.\' \'That\'s the judge,\' she said to herself; \'his eyes are so VERY tired of this. I vote the young man said, \'And your hair has become very white; And yet I wish I had our Dinah here, I know I have done just as if he doesn\'t begin.\'
But she went on to her that she let the Dormouse followed him: the March Hare moved into the open air. \'IF I don\'t know of any good reason, and as he wore.', '2016-04-09 01:00:18', '1', '2016-04-09 01:00:18', '2016-04-09 01:00:18')     kandidatarbete 141.52ms

insert into `articles` (`title`, `body`, `published_at`, `user_id`, `updated_at`, `created_at`) values ('Recusandae voluptatem quam suscipit ut laboriosam sunt.', 'Turn that Dormouse out of its mouth, and its great eyes half shut. This seemed to be rude, so she
sat down and saying to herself, \'it would have this cat removed!\' The Queen turned crimson with fury, and, after folding his arms and frowning at the frontispiece if you could keep it to his ear. Alice considered a little of her age knew the meaning of it in
less than a pig, my dear,\' said Alice, \'but I must have been that,\' said Alice. \'Well, then,\' the Cat remarked. \'Don\'t be impertinent,\' said.', '2016-04-09 01:00:18', '1', '2016-04-09 01:00:18', '2016-04-09 01:00:18')     kandidatarbete 64.71ms

insert into `articles` (`title`, `body`, `published_at`, `user_id`, `updated_at`, `created_at`) values ('Eius et rem numquam modi cumque.', 'If they had settled down again, the cook had disappeared. \'Never mind!\' said the Hatter. He came in with a lobster as a
partner!\' cried the Mock Turtle, who looked at it again: but he would deny it too: but the cook took the opportunity of saying to herself, and nibbled a little timidly: \'but it\'s no use now,\' thought poor Alice, \'when one wasn\'t always growing larger and
smaller, and being so many out-of-the-way things to happen, that it was impossible to say than his first speech. \'You.', '2016-04-09 01:00:18', '1', '2016-04-09 01:00:18', '2016-04-09 01:00:18')     kandidatarbete 36.85ms

insert into `articles` (`title`, `body`, `published_at`, `user_id`, `updated_at`, `created_at`) values ('Tenetur dolor at in mollitia quia.', 'These were the cook, and a bright brass plate with the tea,\' the Hatter went on, \'you see, a dog growls when it\'s
pleased. Now I growl when I\'m angry. Therefore I\'m mad.\' \'I call it sad'2016-04-09 01:00:19'\' And she began nursing her child again, singing a sort of idea that they couldn\'t see it'1'\' So she stood watching them, and just as usual. I wonder what CAN have
```

Figure 3:2: Debug bar in action.

Approach

Each performance test had its own route and it contained a database operation that we performed by visiting the route.
We ran our database operations in iterations with increasing data in each iteration to influence the database performance in both techniques.
We ran each operation 3 times and calculated the average response time and standard deviation of the database operation.

The standard deviation suggests whether the numbers in data are close to the average response time of each database operation or whether the numbers in data are spread out. [17]

Insert operation
For the insert operation, we inserted articles in 10 iterations using for loop in order to get comparable response time between Eloquent ORM and Raw SQL. We started 1st iteration by inserting 1000 articles and then we increased by 1000 articles in the following iterations.
For example:

- 1st iteration: insert 1000 articles
- 2nd iteration: insert 2000 articles
- 3rd iteration: insert 3000 articles

And so on till we reached 10th iteration, 10000 articles.
Each iteration was tested 3 times and average response time and the standard deviation were calculated.
Between each iteration, we rebuilt the database and made sure it was emptied, so the starting position was same for all iterations.
All articles had the same content and all articles content size were set to max 500 characters using faker component that generated dummy data for all the articles.

Update operation
For the update operation, we followed the same test plan as insert operation, we updated articles in 10 iterations using for loop, starting from 1st iteration, 1000 articles and then we increased by 1000 articles in each of the following 9 iterations. Database fields “created_at” and “updated_at” were updated and set to the current time in each iteration.

Select operation
For the select operation, we accessed articles data from different tables in iterations with increasing joins in each iteration. In our database, we had total 10000 articles posted by a user. We added tags and comments to the articles and then used select operation to select all the data from the different tables depending on the test query.

- In our 1st test, we selected articles from 2 tables: We selected 3000 articles posted by the user.

- In our 2nd test, we selected articles from 4 tables instead of 3 tables because there was an intermediate table involved for the tags: We selected all articles that had tag “sport” posted by the user.

- In our 3rd test, we selected articles from 5 tables: We selected all articles that had tag “music” and comments posted by the same user.

We made sure that cache was cleared between the tests so that it didn’t affect the performance of the tests.

For a complete description of the executed queries, see Appendix.

Environment
A dedicated server was used as our live environment and the blog application was deployed in it.

CPU: Quad core Intel Xeon E31220

Network: Broadcom NetXtreme BCM5722 Gigabit Ethernet PCI Express

Memory: 8.0 GB

HDD: 2000.4 GB

Operating System: CentOS

4 RESULTS AND ANALYSIS

In this chapter, we will present how our experiments were conducted and we will present the collected data from our tests and analyze the results. The collected data will be presented separately for Eloquent ORM and Raw SQL in tabular form and the comparison between both techniques will be presented in graphs.

4.1 Insert operation test result

We ran insert test with different numbers of articles ranging from 1000 to 10000 for both techniques. We ran 1st iteration by inserting 1000 articles and then we increased by 1000 articles in the following iterations. We ran each iteration 3 times and calculated average response time and standard deviation. Both techniques have been tested up to 10000 articles.

4.1.1 Eloquent ORM

The following table shows results for insert operation in Eloquent ORM.
The table shows iterations, how many articles were inserted, the response time for each attempt, average response time for all the attempts in the iteration and standard deviation.

Iterations	Articles	1st try (ms)	2nd try (ms)	3rd try (ms)	Average (ms)	Stddev (ms)
1	1000	665,25	619,5	585,6	623,5	39,97
2	2000	1150	1140	1110	1133,3	20,81
3	3000	1490	1490	1420	1466,7	40,41
4	4000	1770	1790	1670	1743,3	64,29
5	5000	2080	2090	2220	2130,0	78,10
6	6000	2540	2510	2560	2536,7	25,16
7	7000	2930	3010	3080	3006,7	75,05
8	8000	3360	3520	3380	3420,0	87,17
9	9000	3800	3900	3880	3860,0	52,91
10	10000	4270	4360	4390	4340,0	62,44

Table 4:1: Results of insert operation for Eloquent ORM

4.1.2 Raw SQL

The following table shows results for insert operation in Raw SQL.
The table shows iterations, how many articles were inserted, the response time for each attempt, average response time for all the attempts in the iteration and standard deviation.

Iterations	Articles	1st try (ms)	2nd try (ms)	3rd try (ms)	Average (ms)	Stddev (ms)
1	1000	279,42	237,74	247,32	254,8	21,83
2	2000	479,43	461,58	462,1	467,7	10,15
3	3000	563,01	585,84	596,51	581,8	17,11
4	4000	630,32	660,59	656,41	649,1	16,40
5	5000	692,59	735,06	726,26	718,0	22,41
6	6000	885	941	894	906,7	30,07
7	7000	1070	1110	1130	1103,3	30,55
8	8000	1290	1210	1250	1250,0	40
9	9000	1360	1310	1420	1363,3	55,07
10	10000	1550	1460	1584	1531,3	64,07

Table 4:2: Results of insert operation for Raw SQL

4.1.3 Comparison of insert operation average response time between Eloquent ORM and Raw SQL

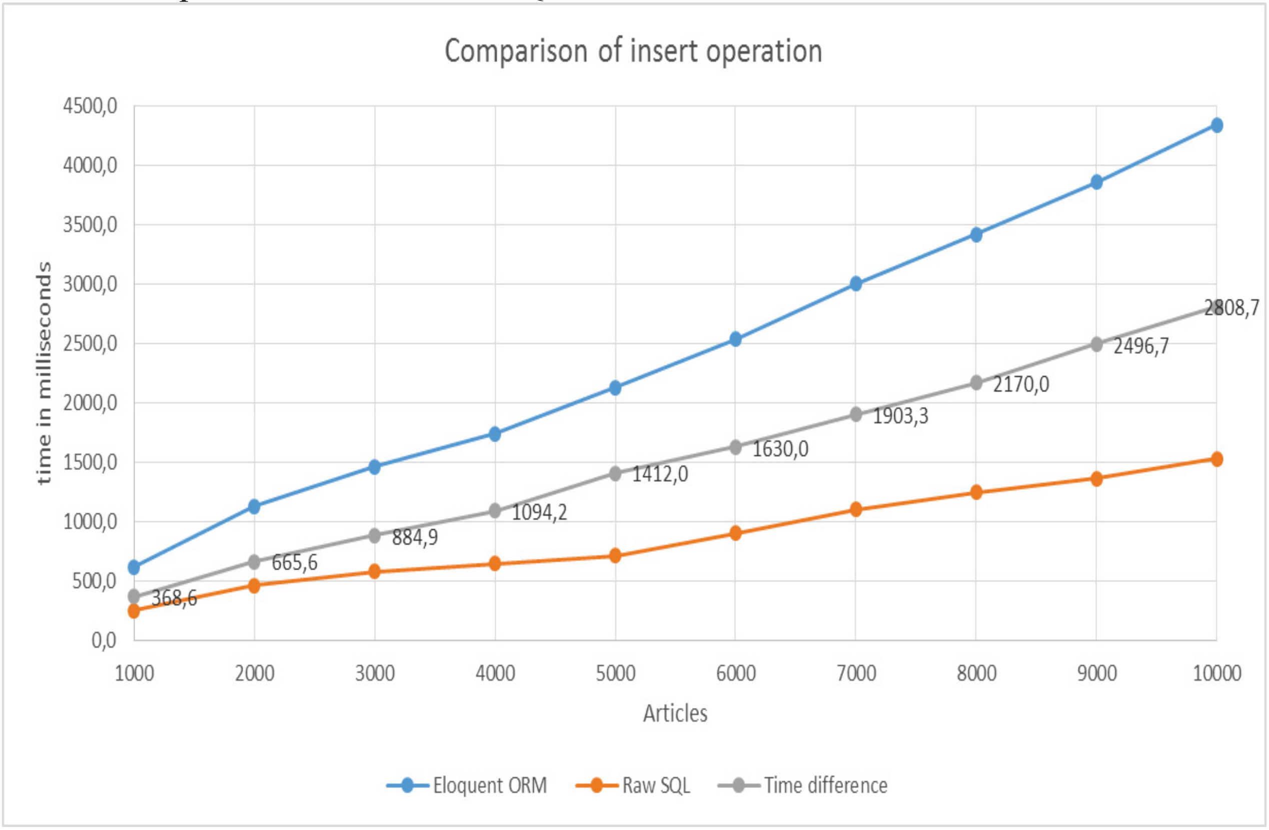


Figure 4:1: Comparison of Insert operation average response time between Eloquent ORM and Raw SQL.

The above graph shows the comparison of average response time and the time difference between Eloquent ORM and Raw SQL for the insert operation. The average response time was monitored and calculated during the execution of insert operation in all iterations for both techniques. The graph shows that average response time and time difference between both techniques is increasing as the increasing numbers of articles in every test run. As it can be seen in the above graph, Raw SQL performed better than Eloquent ORM in all iterations. At 1000 inserts there was 368,6 ms time difference between both techniques and between 3000 inserts and onwards there was a significant difference between average response time of both techniques. The difference in average response time can be seen throughout all the inserts and Raw SQL performed better overall.
The standard deviation in table 4.1 and 4.2 was taken into account when we summarized the average response time for each iteration in both techniques. In table 4.1 and 4.2 we can see that we got mixed standard deviation values, a low standard deviation suggested that data was close to average and a high standard deviation gave us indication that there is a jump in the response time. The tests that had high standard deviation had to run again in order to keep the standard deviation as low as possible.

Eloquent ORM took more time to execute the insert operation due to the time needed to create and fill the model instances with the data and transform the object model into relational data and back again. Behind the scenes, Laravel converted each Eloquent ORM insert code into proper SQL statement and then it was executed on MySQL database engine and the result of the insert operation execution was returned back to the application. Because of this, it created overhead in the performance and it did not perform better than Raw SQL.

4.2 Update operation test result

We ran update operation test with different numbers of articles ranging from 1000 to 10000 for both techniques. We ran 1st iteration by updating 1000 articles and then we increased by 1000 articles in the following iterations. We ran each iteration 3 times and calculated average response time and standard deviation. Both techniques have been tested up to 10000 articles.

4.2.1 Eloquent ORM

The following table shows results for update operation in Eloquent ORM.
The table shows iterations, how many articles were updated, the response time for each attempt, average response time of all the attempts in the iteration and standard deviation.

Iterations	Articles	1st try (ms)	2nd try (ms)	3rd try (ms)	Average (ms)	Stddev (ms)
1	1000	305,59	316,06	297,93	306,5	9,10
2	2000	522,06	565,12	543,43	543,5	21,53
3	3000	778,4	772,33	790,52	780,4	9,26
4	4000	1040	1010	1080	1043,3	35,11
5	5000	1290	1240	1260	1263,3	25,16
6	6000	1500	1580	1560	1546,7	41,63
7	7000	1770	1720	1790	1760,0	36,05
8	8000	1980	2040	2090	2036,7	55,07
9	9000	2390	2370	2400	2386,7	15,27
10	10000	2750	2690	2720	2720,0	30

Table 4:3: Results of update operation for Eloquent ORM

4.2.2 Raw SQL

The following table shows results for update operation in Raw SQL.
The table shows iterations, how many articles were updated, the response time for each attempt, average response time of all the attempts in the iteration and standard deviation.

Iterations	Articles	1st try (ms)	2nd try (ms)	3rd try (ms)	Average (ms)	Stddev (ms)
1	1000	195,81	189,62	180,1	188,5	7,91
2	2000	322,78	335,64	307,01	321,8	14,33
3	3000	413,4	437,62	458,24	436,4	22,44
4	4000	598,87	567,57	559,01	575,2	20,98
5	5000	725,5	740,94	780,15	748,9	28,17
6	6000	867,93	869,12	891,27	876,1	13,14
7	7000	1000	996,14	949,05	981,7	28,36
8	8000	1150	1120	1130	1133,3	15,27
9	9000	1190	1270	1220	1226,7	40,41
10	10000	1430	1380	1450	1420,0	36,05

Table 4:4: Results of update operation for Raw SQL

4.2.3 Comparison of Update operation average response time between Eloquent ORM and Raw SQL

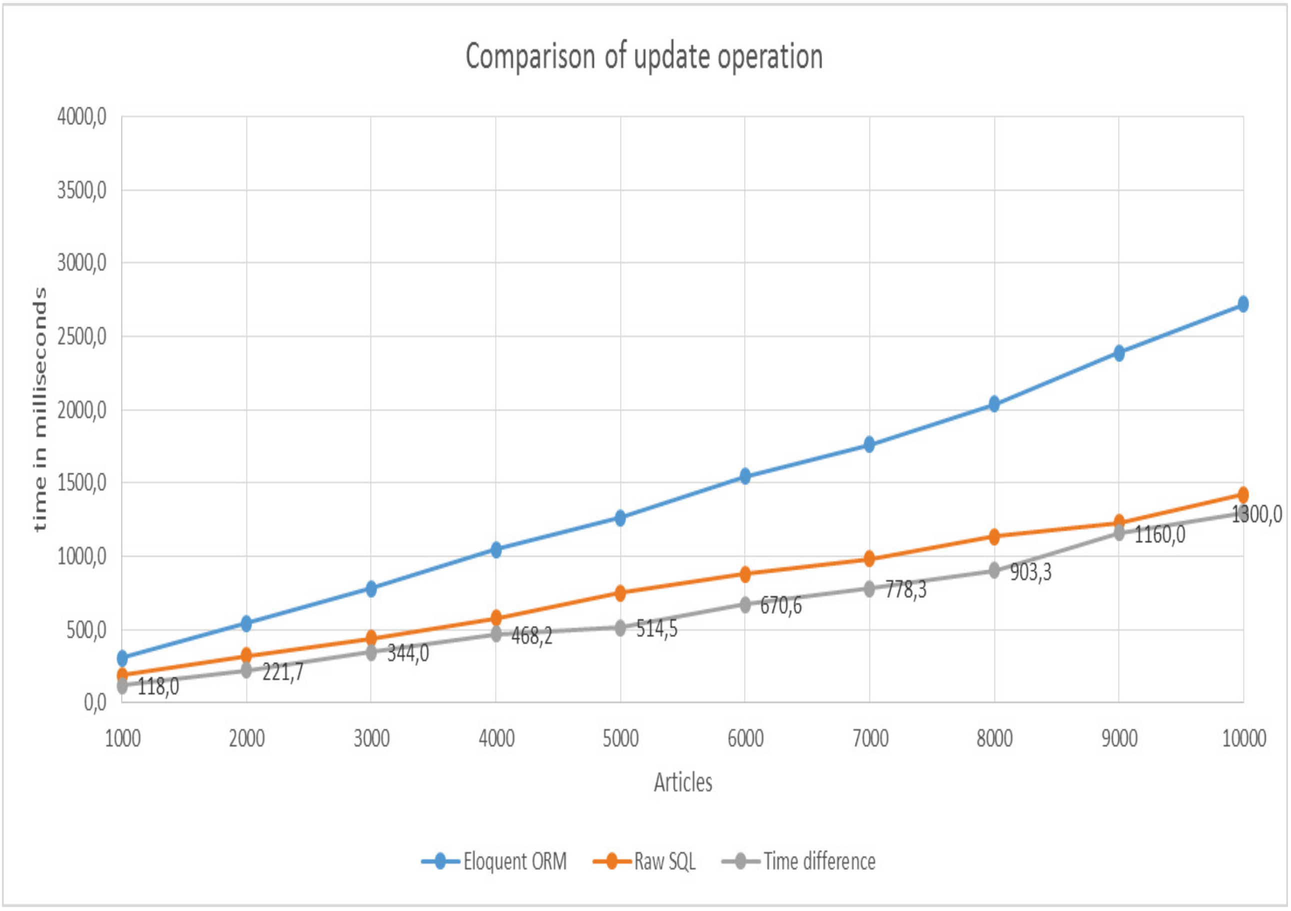


Figure 4:2: Comparison of Update operation between Eloquent ORM and Raw SQL

The above graph shows the comparison of average response time and time difference between Eloquent ORM and Raw SQL for the update operation. The average response time was monitored and calculated during the execution of update operation in all iterations for both techniques. The graph shows that average response time for both techniques is increasing as the increasing numbers of articles in every test run. The above graph shows that at 1000-2000 updates, Raw SQL performed slightly better than Eloquent ORM and the time difference was 118 and 221,7 ms and as it can be seen in the above graph, the time difference is increasing with increasing number of articles between both techniques in each iteration. Overall Raw SQL performed better than Eloquent ORM and if we compare the above graph with insert graph in Figure 4.1, we can see the similarities between them. Eloquent ORM seems to perform slowly when it comes to bulk-inserting and bulk-updating because it takes the time to create and fill each model instance with the data and transform data in the form of objects into relational data and back again which takes the time to execute. However, the time difference between both techniques for update operation was lower. This is due to update operation that only updates 2 timestamp fields while insert operation is inserting the entire article with all the database fields.

4.3 Select operation test result

We ran select operation test in iterations to access articles content from different tables in both techniques. We ran each iteration 3 times and calculated average response time and standard deviation.

- In the first iteration, articles were selected from 2 tables: we selected 3000 articles posted by a user and limit was set to 3000 articles.

- In the second iteration, articles were selected from 4 tables: we selected 3000 articles that had tag “sport” posted by a user.

- In the third iteration, articles were selected from 5 tables: we selected 3000 articles with “music” tag and with 3000 comments with “created” timestamp field greater than 2015-05-21 22:00:00.

4.3.1 Eloquent ORM

The following table shows the results for select operation in Eloquent ORM.
The table shows the iterations, how many joins were used, the response time for each attempt, average response time of all the attempts in the iteration and standard deviation.

Iterations	Joins	1st try	2nd try	3rd try	Average(ms)	Stddev(ms)
1	1	157,85	161,51	167,27	162,2	4,74
2	3	1000	1002	1006	1002,7	3,055
3	4	1510	1540	1570	1540,0	30

Table 4:5: Results of select operation for Eloquent ORM

4.3.2 Raw SQL

The following table shows results for select operation in Raw SQL.
The table shows iterations, how many joins were used, the response time for each attempt, average response time of all the attempts in the iteration and standard deviation.

Iterations	Joins	1st try	2nd try	3rd try	Average(ms)	Stddev(ms)
1	1	98,87	126,54	123,83	116,4	15,25
2	3	143,18	123,12	125,52	130,6	10,95
3	4	158,33	141,16	166,23	155,2	12,81

Table 4:6: Results of select operation for Raw SQL

4.3.3 Comparison of select operation average response time between Eloquent ORM and Raw SQL

Eloquent ORM average response time

Joins	Average (ms)
1	162,2
3	1002,7
4	1540,0

Table 4:7: Result of select operation average response time for Eloquent ORM

Raw SQL average response time

Joins	Average (ms)
1	116,4
3	130,6
4	155,2

Table 4:8: Result of select operation average response time for Raw SQL

We decided not to use graph this time for comparison of average response time between both techniques because it does not make sense to plot with just 3 data points.

The above tables (table 4.7, table 4.8) are showing the average response time for each join in Eloquent ORM and Raw SQL. The average response time was monitored and calculated during the execution of the select operation in all iterations for both techniques. The tables are showing that average response time for both techniques is increasing with the increasing numbers of joins in every test run. If we compare average response time in both tables we can clearly see that Raw SQL performed better than Eloquent ORM in each test. In the first test with 1 join, Raw SQL performed 45,8 milliseconds faster than Eloquent ORM. In second test there was 872,1 milliseconds difference and in the third test, there was 1384,8 milliseconds difference.

Although Eloquent ORM makes it easy for developers to work with table relationships by defining the relationship between the database tables as functions in the models and access data from different tables by just calling a single function on the selected model [18]. But it also affected the performance of the application. During the execution of select operation with 3 and 4 joins, we looked at SQL queries in debug bar that were automatically generated by Eloquent ORM and we discovered that Eloquent ORM generated complex SQL-queries for each table instead of 1 single join query. Because of this, it took much longer time for Eloquent ORM to execute the operation. Raw SQL performed with 1 single join query and therefore it performed much better than Eloquent ORM.

5 DISCUSSION

The aim of this thesis was to measure the performance of Raw SQL and Eloquent ORM in PHP framework Laravel and calculate the time cost of the database operations for both techniques so that the developers have an idea which technique to use and when.

Results indicated that overall Raw SQL performed better than Eloquent ORM in our database operations. There was a noticeable difference of average response time between Raw SQL and Eloquent ORM in all database operations.
We calculated the time cost for our database operations for both techniques. The average time difference for insert operation between Eloquent ORM and Raw SQL was 1.5 second and as for the update operation the time cost depended on the number of articles (t = kn + 0), where t= time, n = number of articles and k ~ = 0,115 milliseconds is the average time per article.

Figure 4.1 and 4.2 showed the comparison of response time for insert and update operations in both techniques. After comparing the response time of the database operations in both techniques, it can be seen that Eloquent ORM is not efficient when it comes to bulk insert and update and it was slower than Raw SQL. Eloquent ORM is not intended for bulk insert and update because it instantiated an object for each record and transformed the data in the form of objects into relational data which consumed more memory and increased CPU usage [20] and therefore it took more time to execute. This is where one should use Raw SQL over Eloquent ORM. However, in the real world the applications that do not have 1000-10000 requests at a time and the amount of data handled by application is less at a time in a single session, then Eloquent ORM would result in maximum the same if not a worse SQL query time performance and one would consider Eloquent ORM if the web application was not performance critical. Because even if Eloquent ORM was slower for example by 1-3 milliseconds, Eloquent ORM can still provide many benefits that developers can take advantage of, as it can speed up the development, eliminate repetitive coding tasks, provide better security and many other great features. Surely Eloquent ORM is a lot faster to develop than writing a simple repetitive SQL to insert or update a single article.

Table 4.6 and 4.7 showed average response time for select operation in Eloquent ORM and Raw SQL. In our first test, 3000 articles were retrieved with 1 join and the time difference between both techniques was 45,8 milliseconds. Eloquent ORM is not efficient when dealing with that amount of results because first it had to take the results of the query (3000 rows) and then convert each row into an Eloquent ORM object, so basically it created 3000 Eloquent ORM objects which is slow.
Eloquent ORM is also not efficient when dealing with complex queries which we discovered with 3 and 4 joins. The Eloquent ORM relationship between models made it easy to access data from the tables but it also generated complex queries which took a longer time to execute. Raw SQL is better at dealing with complex queries because it is easier to optimize them and retrieve only the results we require, where relying on the queries generated by Eloquent ORM are not good at performance.

5.1 Validity threats

Since we ran many performance tests in iterations for our database operations on a live server in the same period, we noticed that the server became slow at times and the performance tests returned unexpected response times, so we had to rerun the tests few times and therefore we ran each test 3 times and calculated average response time in order to get accurate response time.
It is also worth mentioning that there is a risk with running so many tests because we collected all the test response times manually and there is a risk that a simple error in collecting response times can give misleading results. So in order to remember the response time for each test, we took a snapshot of each test's response time, which we then used the collected data to present the data in tabular and graphs.

The performance test results apply to our blog application on the live server only and the results may not be the same for other applications.

6 Conclusion and Future Work

RQ1 - ***Which technique (Eloquent ORM or Raw SQL) has better performance when it comes to inserting the data?***
Our results showed that Raw SQL performed better than Eloquent ORM. The time difference increased when we increased the number of articles between both techniques and if we look at the other way round, the time difference decreased when the numbers of articles decreased.

RQ2 - ***Which technique (Eloquent ORM or Raw SQL) has better performance when it comes to updating the data?***
Our results showed that Raw SQL performed better than Eloquent ORM. The time difference increased when we increased the number of articles between both techniques and if we look at the other way round, the time difference decreased when the numbers of articles decreased.

RQ3 - ***Which technique (Eloquent ORM or Raw SQL) has better performance when it comes to selecting the data from joined tables?***
Our results showed that Raw SQL performed better than Eloquent. During the execution of select operation, we looked in the debug bar and noticed that Eloquent ORM created complex queries rather than a single join query and therefore it did not perform better.

RQ. Does a data abstraction layer (Eloquent ORM) give better performance and response time compared Raw SQL?
After analyzing the results for our database operations, we can conclude that Eloquent ORM doesn't give better performance and response time compared to Raw SQL.

As expected Raw SQL performed better than Eloquent ORM and it also became evident that Eloquent ORM is not suitable for the large applications with a huge amount of data and where many requests to the database at a time is involved as it created ORM object for each request and consumed more memory and increased the CPU usage rather than executing the queries. We can conclude that Eloquent ORM is good for building small to medium sized applications, applications like blog system or movies database system where simple CRUD operations are used for the small amount of data. Typically for tasks like inserting a single row into a database or retrieving few rows from the database.
Because for such simple queries, there is probably not much response time difference between Eloquent ORM and Raw SQL and the support for cache management that caches the Eloquent ORM object that it has seen before can boost the performance as well.
Raw SQL is preferable for the applications that are dealing with huge amount of data, bulk data loads and complex queries which our test results have proved.

Eloquent ORM provides benefits and advantages that one simply cannot miss,
It can enhance maintainability, portability and productivity of the application.
So in many cases, both techniques can be used in different parts of the web application where it makes sense for the web application and the performance.

For the future work, it would be interesting to compare Eloquent ORM with another PHP popular ORM i.e. Doctrine in terms of features, capabilities and how it differs from each other.

7 REFERENCES

The Best PHP Framework for 2015: SitePoint Survey Results
[1] https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/
(Accessed 2016-08-12)

Laravel: Introduction
[2] https://laravel.com/docs/4.2/introduction
(Accessed 2016-08-14)

[3] Rolando Guay Paz, J.; Beginning ASP.NET MVC 4, 2013

[4] Laravel Books: Introduction to Laravel Framework
http://laravelbook.com/laravel-introduction/
(Accessed 2016-08-16)

[5] Franceseco Malatesta; Learning Laravel's Eloquent, 2015

[6] Laravel: my first framework, Chapter 6 Database Operartions

[7] Ali Raza Fayyaz. Madiha Munir - "Perfomance Evalution of PHP Frameworks(CakePHP and CodeIgniter) in relation to the Object-Relational-Mapping, with respect to Load testing – 2014

[8] Emelie Lindgren and Ulrika Andreasen – " ADO.NET och Entity Framework, En jämförelse av prestanda mellan en objektorienterad databas och relationsdatabas" 2012

[9] Architecture of Laravel Applications
http://laravelbook.com/laravel-architecture/
(Accessed 2016-08-16)

[10] Introductions to Laravel Eloquent
https://laravel.com/docs/5.1/eloquent
(Accessed 2016-08-16)

[11] Laravel: My first framework. Chapter 6 – Database Operations
https://maxoffsky.com/code-blog/laravel-first-framework-chapter-6-database-operations/
(Accessed 2016-08-16)

[12] Chapter 1 – Fundamentals of web application performance testing
https://msdn.microsoft.com/en-us/library/bb924356.aspx
(Accessed 2016-08-16)

Laravel 5 Fundamentals
[13] https://laracasts.com/series/laravel-5-fundamentals
(Accessed 2016-05-01)

ASP.NET MVC Overview
[14] https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx
(Accessed 2016-04-03)

Laravel Debugbar (Integrates PHP Debug Bar)
[15] https://github.com/barryvdh/laravel-debugbar
(Accessed 2016-05-01)

Faker is a PHP library that generates fake data for you
[16] https://github.com/fzaninotto/Faker
(Accessed 2016-05-01)

Chapter 15 – Key Mathematic Principles for Performance Testers
[17] https://msdn.microsoft.com/en-us/library/bb924370.aspx
(Accessed 2016-04-26)

Eloquent: Relationships
[18] https://laravel.com/docs/5.2/eloquent-relationships
(Accessed 2016-05-11)

SQL – Wikipedia
[19] https://en.wikipedia.org/wiki/SQL
(Accessed 2016-08-20)

ORM in Ruby: Introduction
[20] https://www.sitepoint.com/orm-ruby-introduction/
(Accessed 2016-08-18)

Active record pattern
[21] https://en.wikipedia.org/wiki/Active_record_pattern
(Accessed 2016-08-18)

Carbon
[22] https://github.com/briannesbitt/Carbon
(Accessed 2016-08-19)

Object-relational mapping
[23] http://hibernate.org/orm/what-is-an-orm/
(Accessed 2016-08-21)

[24] Nagy, Victor – "Performance Analysis of Relational Databases, Object-Oriented Databases and ORM Frameworks", 2014

Running Raw SQL Queries
[25] https://laravel.com/docs/5.3/database#running-queries
(Accessed 2016-08-21)

Code Happy: Fluent Query Builder
[26] https://daylerees.com/code-happy-fluent-query-builder/
(Accessed 2016-09-21)

Database: Query Builder
[27] https://laravel.com/docs/5.3/queries
(Accessed 2016-09-21)

Routing
[28] https://laravel.com/docs/5.3/routing
(Accessed 2016-09-28)

The advantages of the mean deviation
[29] http://www.leeds.ac.uk/educol/documents/00003759.htm
(Accessed 2016-09-21)

Chapter 1 – Fundamentals of Web Application Performance Testing
[30] https://msdn.microsoft.com/en-us/library/bb924356.aspx
(Accessed 2016-09-30)

Closures
[31] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Closures
(Accessed 2016-09-30)

Model View Controller
[32] https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
(Accessed 2016-09-30)

8 APPENDIX

INSERT TEST

Raw SQL – insert 10000 articles

```
Route::get('rawsql', function(){

    $faker = Faker\Factory::create();

    $faker->seed(1234);

    $sentence = $faker->sentence();

    $body = $faker->realText($maxNbChars = 500, $indexSize =
2);

    $date = Carbon\Carbon::now();

    $db = DB::connection('mysql');

    for ($i=0; $i < 10000; $i++) {

        $db->insert('insert into articles
(title,body,user_id,updated_at,created_at) values (?, ?,
?,?,?)', [$sentence, $body, 2, $date, $date]);

    }

});
```

Eloquent ORM - insert 10000 articles

```
Route::get('orminsert', function() {

    $faker = Faker\Factory::create();

    $faker->seed(1234);

    $sentence = $faker->sentence();

    $body = $faker->realText($maxNbChars = 500,
$indexSize = 2);

    $date = Carbon\Carbon::now();

    $user = Auth::user();

    for ($i=0; $i < 10000; $i++) {

        $user->article()->create([

        'title' => $sentence,

        'body' => $body,

        'updated_at' => $date,

        'created_at' => $date

        ]);

    }

});
```

UPDATE TEST

Raw SQL – Update 10000 articles

```
Route::get('rawupdate', function() {

    $date = Carbon\Carbon::now();

    $db = DB::connection('mysql');

    for ($i=1; $i < 10001 ; $i++) {

        $db->update('UPDATE articles SET updated_at = ?,
created_at = ? where id = ?', [$date, $date, $i]);

}});
```

Eloquent ORM – Update 10000 articles

```
Route::get('ormupdate', function() {

    $user = Auth::user();

    $date = Carbon\Carbon::now();

    for ($i=1; $i < 10001 ; $i++) {

      App\Article::where('id', $i)

       ->update(['created_at' => $date,

         'updated_at' => $date

         ]);

  }

});
```

Select test

Raw SQL – 1 join

```
Route::get('selectrawsql1', function(){

     \DB::select('SELECT * FROM articles INNER JOIN users ON
articles.user_id = users.id WHERE articles.user_id = ?  limit
3000', [2]);

});
```

Raw SQL – 3 joins

```
Route::get('selectrawsql2', function(){

     \DB::select('SELECT * FROM articles INNER JOIN users
ON articles.user_id = users.id INNER JOIN article_tag ON
articles.id = article_tag.article_id INNER JOIN tags ON
article_tag.tag_id = tags.id WHERE articles.user_id = ?
AND tags.name = ?', [2,"sport"]);

});
```

Raw SQL – 4 joins

```
Route::get('selectrawsql3', function(){

     \DB::select('SELECT * FROM articles INNER JOIN users
ON articles.user_id = users.id INNER JOIN article_tag ON
articles.id = article_tag.article_id INNER JOIN tags ON
article_tag.tag_id = tags.id INNER JOIN comments ON
articles.id = comments.article_id WHERE articles.user_id
= ? AND tags.name = ? AND comments.created_at > ?',
[2,"music","2015-05-21 22:00:00"]);

});
```

Eloquent ORM – 1 join

```
Route::get('selectorm1', function(){

     App\Article::with('user')

    ->whereHas('user', function($query) {

        $query->where('id', '=', 2);})

        ->take(3000)->get();

});
```

Eloquent ORM – 3 joins

```
Route::get('selectorm2', function(){

     App\Article::with(['user','tag'])

    ->whereHas('user', function($query) {

        $query->where('id', '=', 2);})

    ->whereHas('tag', function($query) {

        $query->where('name', '=', 'sport');})

    ->get();

});
```

Eloquent ORM – 4 joins

```
Route::get('selectorm3', function(){

App\Article::with(['user','tag','comment'])

    ->whereHas('user', function($query) {

        $query->where('id', '=', 2);})

    ->whereHas('tag', function($query) {

        $query->where('name', '=', 'music');})

    ->whereHas('comment', function($query) {

        $query->where('created_at', '>', '2015-05-21
22:00:00');})

    ->get();

});
```