JSON FAILA KĀ DATU GLABĀŠANAS VEIDA PIELIETOŠANAS PRIEKŠROCĪBAS

THE ADVANTAGES OF USING A JSON FILE AS A DATA STORAGE METHOD

Authors: **Artis Laizāns**, e-mail: al20142@edu.rta.lv
Scientific supervisors: **Jurijs Musatovs**, e-mail: jurijs.musatovs@rta.lv
Rēzeknes Tehnoloģiju akadēmija
Atbrīvošanas aleja 115, Rēzekne, Latvija

Abstract. *The aim of this work is to showcase adventages of utilizing JSON (JavaScript Object Notation) files as a data storage method which outperforms tradditional MySQL database when there are hundreds and thousands of records of data resulting to shorter load times and less extra load to MySQL database.*

Keywords: *MySQL, JSON, speed, database, loading, load.*

Ievads

Bieži vien kad lietotājs apmeklē kādu mājaslapu, katru reizi tiek veikts pieprasījums uz datubāzi. Piemēram internet veikals, katru reizi kad lietotājs izvēlas kādu preci pievienot grozam, šī informācija tiek saglabāta datubāze, tādejādi ja lietotājs aizver lapu vai atgriežas vēlreiz informācija tiek nolasīta no datubāzes un tiek ielādēta un attēlota lietotājam. Ja katru dienu mājaslapu apmeklē 1000 lietotāju un katru dienu vizmas 100 lietotāji pievieno kaut vienu preci grozam, tad ierakstu skaits datubāze palielinās, un katru reizi kad ir nepieciešams atrast kādu konkrētu ierakstu, datu nolasīšanas laiks kļūst lēnāks.

Šī pētījuma **mērķis** ir noteikt datu nolasīšanas ātrumu no datubāzes un JSON faila, pēc tam analizēt un salīdzināt rezultātus pie 1,100,100,10000 ierakstu skaita gan JSON gan MySQL un aprēķināt vidējo laiku pie 10 testiem.

Tehnoloģijas

- *XAMPP* - programmatūras pakotne lai izveidotu un darbinātu web serveri lokāli;
- *PHP* - serverpusē izpildāma skriptu valoda, kas galvenokārt tiek izmantota web izstrādē;
- *MYSQL* - datubāzes pārvaldības sistēma, ko izmanto daudzi uzņēmumi un organizācijas visā pasaulē;
- *JSON* - viegli lasāms datu apmaiņas formāts. Tas ir populārs veids, kā pārsūtīt un saglabāt strukturētus datus;

Eksperimenta process

Vispirms bija nepieciešams izstrādāt testēšanas vidi ar kuras palīdzību varētu veikt dažādas darbības kā: veikt testu, ierakstu izveide, testa atiestatīšanā un iegūto datu attēlošana un aprēķināšana (1. attēls). Testa vide sastāv no 2 failiem, 1. fails ir lietotāja saskarne un *JavaScript* funkcijas un 2. fails ir *PHP* fails ar klasi priekš funkciju izpildes. Šāda arhitektūra ļauj gan attēlot gan pārvaldīt testa vidi bez nepieciešamības atkārtoti ielādēt lapu kad tiek izsaukta kāda no darbībām. Piemēram pēc "Generate New Records" pogas nospiešanas, izmantojot *AJAX*, informācija tiek nosūtīta uz *PHP* failu kas pēc tam izveido attiecīgo ierakstu skaitu un to atgriez atpakaļ lietotājam un "Total Records Generated" mainās uz tagadējo skaitu. Pēc pogas "Test" nospiešanas *PHP* atgriez laiku kas bija nepieciešams gadījuma ieraksta nolasīšanai no datubāzes un JSON un katrs laiks tiek saglabāts atsevišķā mainīgajā kas katru reizi saskaitās kopā un tiek dalīts ar "Total Records Tested" vērtību lai aprēķinātu vidējo laiku.

https://doi.org/10.17770/het2024.28.8261

	MySQL	JSON
Time	0.00021409988403320312	0.00022292137145996094
Average Time	0.00021409988403320312	0.00022292137145996094
Data	{"account":"1","property":"1","services":[{"id":"example","data":"test"},{"id":"example","data":"test"},{"id":"example","data":"test"}]}	{"account":1,"property":"1","services":[{"id":"example","data":"test"},{"id":"example","data":"test"},{"id":"example","data":"test"}]}

Total Records Generated	1
Total Records Tested	1

Generate New Records: 1 [Generate New Records] [Reset Test] [Test]

Test Nr.	MySQL	JSON
1.	0.00021409988403320312	0.00022292137145996094

1. attēls. Testa vide

Testa ierakstu izveidošana

Lai iegūtu maksimāli precīzus datus, nolēmu izmantot identisku datu kopiju kas tiktu ievietota gan datubāzē gan saglabāta JSON failā, dinamiski mainās tikai viena vērtība kas ir ID, un JSON faila nosaukums ir dinamiskais ID un tā paplašinājums. Katru reizi, kad tiek funkcija tiek izsaukta, visi iepriekšējie dati tiek izdzēsti.

```php
public function generateData($count){
    $host = 'localhost';
    $username = 'root';
    $password = '';
    $database = 'test';

    $conn = mysqli_connect($host, $username, $password, $database);
    if (!$conn) {
        die("Connection failed: " . mysqli_connect_error());
    }

    $host = $_SERVER['DOCUMENT_ROOT'];
    array_map('unlink', glob($host.'/json/*'));

    $sql = "DELETE FROM test";
    mysqli_query($conn, $sql);

    for ($i=1;$i<=$count;$i++){
        $json = array();
        $json['account'] = $i;
        $json['property'] = '1';
        $json['services'] = array(
            array('id'=>'example','data'=>'test'),
            array('id'=>'example','data'=>'test'),
            array('id'=>'example','data'=>'test')
        );

        $sql = "INSERT INTO test (account, property, services)
        VALUES (".$json['account']."', ".$json['property']."', '".json_encode($json['services'])."')";
        mysqli_query($conn, $sql);

        file_put_contents($host.'/json/'.$i.'.json', json_encode($json));
    }
    $this->records = $count;
    mysqli_close($conn);
}
```

2. attēls. Datu pievienošana.

Testa ierakstu nolasīšana un ātruma aprēķināšana

Pēc pogas “Test” nospiešanas tiek izpildīts sekojošais kods (3. attēls). Sākuma tiek izvēlēts gadījuma ieraksts kas tiks meklēts gan datubāzē gan kā JSON fails, pēc tam katra funkcija atgriež meklējamo ierakstu (jau gatavā nolasāmā formātā) un laiku kas tika patērēts ieraksta atrašanai. Tālāk katra funkcija tiek paradīta atsevišķi (attēls 4. un 5.)

```php
$test = new test();
$response = new stdClass;

if (isset($_GET['action'])){
    if ($_GET['action'] == 'records'){
        $response->records = $test->records;
    }elseif ($_GET['action'] == 'test'){
        $record = $test->randomRecord();
        list($mysqlData, $mysqlTime) = $test->retrieveFromMySQL($record);
        list($jsonData, $jsonTime) = $test->retrieveFromJSON($record);

        $response->mysql_time = $mysqlTime;
        $response->json_time = $jsonTime;
        $response->mysql_data = $mysqlData;
        $response->json_data = $jsonData;
    }elseif ($_GET['action'] == 'generate'){
        if (isset($_GET['records'])){
            if (is_numeric($_GET['records'])){
                $test->generateData($_GET['records']);
                $response->records = $_GET['records'];
            }
        }
    }
}

$response = json_encode($response);
die($response);
```

3. attēls. Funkciju izpilde un datu atgriešana

```php
public function retrieveFromMySQL($value) {
    $host = 'localhost';
    $username = 'root';
    $password = '';
    $database = 'test';

    $conn = mysqli_connect($host, $username, $password, $database);
    if (!$conn) {
        die("Connection failed: " . mysqli_connect_error());
    }

    $start_time = microtime(true);
    $query = "SELECT * FROM test WHERE account=$value";
    $result = mysqli_query($conn, $query);
    if (mysqli_num_rows($result) > 0) {
       $row = mysqli_fetch_assoc($result);
    }
    $end_time = microtime(true);
    mysqli_close($conn);

    $row['services'] = json_decode($row['services']);
    return [$row, $end_time - $start_time];
}
```

4. attēls. Datu nolasīšana no datubāzes

```php
public function retrieveFromJSON($value) {
    $host = $_SERVER['DOCUMENT_ROOT'];

    $start_time = microtime(true);
    if (file_exists($host.'/json/'.$value.'.json')){
        $data = file_get_contents($host.'/json/'.$value.'.json');
    }
    $end_time = microtime(true);
    return [json_decode($data, true), $end_time - $start_time];
}
```

5. attēls. Datu nolasīšana no JSON faila

3. Testu veikšana un datu apkopošana

Sākuma tika veikts tests lai noteiktu vidējo maksimālo ātrumu kad datubāzē ir tikai 1 ieraksts, un 1 JSON fails (6. attēls). Pēc tam tikai veikti pārējie testi (7,8,9 attēls).

	MySQL	JSON
Time	0.00011897087097167969	0.00013494491577148438
Average Time	0.00018651485443115233	0.0001650571823120117
Data	{"account":"1","property":"1","services":[{"id":"example","data":"test"},{"id":"example","data":"test"},{"id":"example","data":"test"}]}	{"account":1,"property":"1","services":[{"id":"example","data":"test"},{"id":"example","data":"test"},{"id":"example","data":"test"}]}

Total Records Generated	1
Total Records Tested	10

Generate New Records: 1 | Generate New Records | Reset Test | Test

Test Nr.	MySQL	JSON
10.	0.00011897087097167969	0.00013494491577148438
9.	0.00023484230041503906	0.00016188621520996094
8.	0.00026512145996093750	0.00017499923706054688
7.	0.00023198127746582030	0.00016283988952636720
6.	0.00017118453979492188	0.00016689300537109375
5.	0.00014400482177734375	0.00014591217041015625
4.	0.00013208389282226562	0.00013899803161621094
3.	0.0001678466796875	0.00018811225891113280
2.	0.00025010108947753906	0.00016808509826660156
1.	0.00014901161193847656	0.00020790100097656250

6. attēls. 1 ieraksts 10 testi

	MySQL	JSON
Time	0.0002567768096923828	0.00021910667419433594
Average Time	0.00026607513427734375	0.00022461414337158204
Data	{"account":"20","property":"1","services":[{"id":"example","data":"test"},{"id":"example","data":"test"},{"id":"example","data":"test"}]}	{"account":20,"property":"1","services":[{"id":"example","data":"test"},{"id":"example","data":"test"},{"id":"example","data":"test"}]}

Total Records Generated	100
Total Records Tested	10

Generate New Records: 100 | Generate New Records | Reset Test | Test

Test Nr.	MySQL	JSON
10.	0.0002567768096923828	0.00021910667419433594
9.	0.00023603439331054688	0.0002281665802001953
8.	0.0003170967102050781	0.00024008750915527344
7.	0.00023603439331054688	0.0002231597900390625
6.	0.00023698806762695312	0.00020503997802734375
5.	0.0002608299255371094	0.0002281665802001953
4.	0.0003509521484375	0.0002510547637939453
3.	0.0002589225769042969	0.00021314620971679688
2.	0.00023412704467773438	0.0002181529998779297
1.	0.00027298927307128906	0.0002200603485107422

7. attēls. 100 ieraksti 10 testi

	MySQL	JSON
Time	0.0007798671722412109	0.000225067138671875
Average Time	0.0007911443710327149	0.00021963119506835939
Data	{"account":"618","property":"1","services":[{"id":"example","data":"test"},{"id":"example","data":"test"},{"id":"example","data":"test"}]}	{"account":618,"property":"1","services":[{"id":"example","data":"test"},{"id":"example","data":"test"},{"id":"example","data":"test"}]}

Total Records Generated	1000
Total Records Tested	10

Generate New Records: 1000 | Generate New Records | Reset Test | Test

Test Nr.	MySQL	JSON
10.	0.0007798671722412109	0.000225067138671875
9.	0.0008280277252197266	0.0002200603485107422
8.	0.00078582763671875	0.0002231597900390625
7.	0.0007607936859130859	0.00022292137145996094
6.	0.0007929801940917969	0.00021910667419433594
5.	0.000782012939453125	0.00021696090698242188
4.	0.0007679462432861328	0.000225067138671875
3.	0.0007801055908203125	0.00020003318786621094
2.	0.000782012939453125	0.00021195411682128906
1.	0.0008518695831298828	0.0002319812774658203

8. attēls. 1000 ieraksti 10 testi

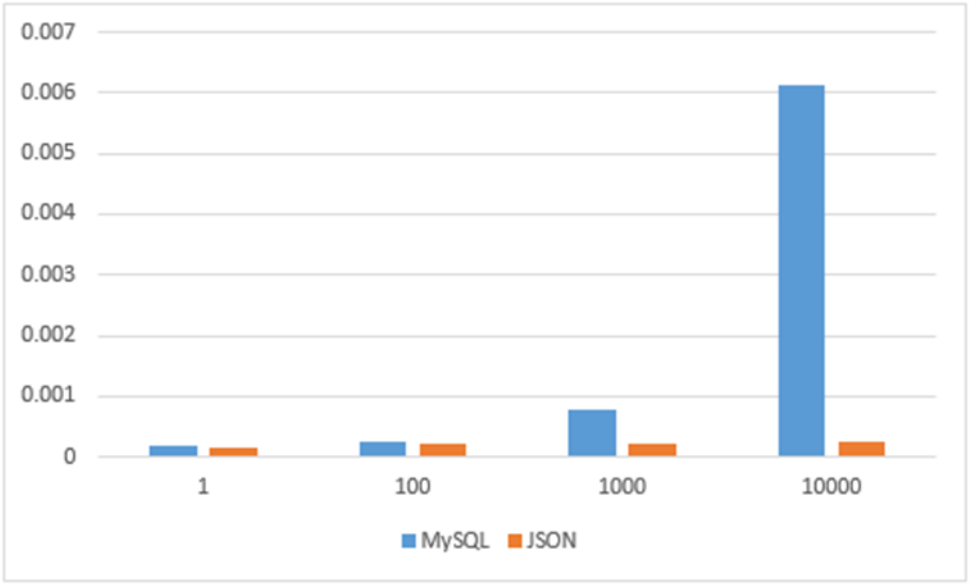
	MySQL	JSON
Time	0.005733013153076172	0.00024390220642089844
Average Time	0.00613400936126709	0.00024950504302978516
Data	{"account":"6091","property":"1","services": [{"id":"example","data":"test"}, {"id":"example","data":"test"}, {"id":"example","data":"test"}]}	{"account":6091,"property":"1","services": [{"id":"example","data":"test"}, {"id":"example","data":"test"}, {"id":"example","data":"test"}]}

Total Records Generated	10000
Total Records Tested	10

Generate New Records: 10000 | Generate New Records | Reset Test | Test

Test Nr.	MySQL	JSON
10.	0.005733013153076172	0.00024390220642089844
9.	0.007354021072387695	0.00024199485778808594
8.	0.006660938262939453	0.00025606153539550781
7.	0.005751132965087891	0.00024914741516113328
6.	0.005934953689575195	0.00026702880859375
5.	0.00671696662902832	0.000244140625
4.	0.005727052688598633	0.00025892257690429969
3.	0.005861043930053711	0.00025391578674316406
2.	0.005810976028442383	0.00023293495178222656
1.	0.005789995193481445	0.00024700164794921875

9. attēls. 10000 ieraksti 10 testi



10. attēls. Grafiski attēlots datu nolasīšanas ātrums

Secinājumi

Kā redzams 10. attēlā, datu nolasīšana no JSON faila vienmēr ir daudz ātrāka nekā no datubāzes. Jo vairāk ir ierakstu jo ātrāk tos būs nolasīt no JSON faila nekā no datubāzes, piemēram ja ir nepieciešams saglabāt groza informāciju par katru lietotāju un vidēji dienā veikalu apmeklē 10,000 lietotāju, un ja 1,000 lietotāju veic pasūtījumu vai jebkādu darbību ar grozu kura būtu jāsaglabā tad izdevīgi to ir darīt JSON failā, tādejādi ietaupot datubāzes resursus un paātrinot informācijas nolasīšanu. Šai metodei ir arī sliktā puse, ja piemēram ir nepieciešams saglabāt svarīgu vai privātu informāciju tad obligāti jānodrošina ierobežota piekļuve JSON failu mapei, un ja tas netiek pilnība vai pareizi izdarīts informācija var tikt nopludināta, tāpēc tādā gadījuma glabāt informāciju datubāze ir labāka un drošāka izvēle.

Uzskatu ka šis paņēmiens ir diezgan noderīgs situācijās kur informācijas nav privāta vai svarīga un ja katru dienu tiek veikti simtiem vai tūkstošiem jaunu ierakstu.

Summary

As seen in picture 10. reading data from a JSON file is always much faster than from a database. The more records there are, the faster they will be read from a JSON file compared to a database. For example, if it's necessary to store cart information for each user and on average 10,000 users visit the store per day, and if 1,000 users make an order or any action with the cart that needs to be saved, then it's advantageous to do so in a JSON file, thus saving database resources and speeding up information retrieval. However, this method also has its downsides. For instance, if it's necessary to store important or private information, restricted access to the JSON file directory must be ensured. If this is not done properly, information could be compromised, making storing the information in a database a better and more secure choice in such cases. I believe this approach is quite useful in situations where the information is not private or critical, and hundreds or thousands of new records are being made every day.

Literatūra

1. XAMPP https://en.wikipedia.org/wiki/XAMPP
2. JSON https://en.wikipedia.org/wiki/JSON
3. AJAX https://en.wikipedia.org/wiki/Ajax_(programming)
4. file_get_contents https://www.php.net/manual/en/function.file-get-contents.php
5. MySQL https://en.wikipedia.org/wiki/MySQL
6. PHP https://en.wikipedia.org/wiki/PHP
7. JavaScript https://developer.mozilla.org/en-US/docs/Web/JavaScript