УДК 621.37:637.142.

Брацький В.О.
Національний університет харчових технологій

ОБРОБЛЕННЯ Й АНАЛІЗ LOG-ФАЙЛІВ ЗА ДОПОМОГОЮ ПРОГРАМНОГО ПРОДУКТУ LOGHELPER

У статті розглянуто програмний продукт, який працює з log-файлами, що є незрозумілими для простого користувача. Запропоновано метод збереження log-файлів, їх аналізу та прийняття рішень щодо виправлень виявлених помилок користувача в розподіленій системі. Описано структуру log-файлів, з'ясовано їхнє призначення. Розглянуто формат JSON і доведено, чому саме в цьому форматі доцільно описувати логи. Також визначено алгоритм аналізу й збереження файлів за допомогою GridFS в MongoDB. Після оброблення даних запропоноване вирішення виявлених відмов у системі. За допомогою програмного продукту проведений порівняльний аналіз параметрів збереження й оброблення log-файлів у реляційній (SQLServer) і нереляційній (MongoDB) базах даних.

Ключові слова: *бази даних, SQL, NoSQL, JSON, реляційні системи керування базами даних (далі – СКБД), нереляційні СКБД, MongoDB, SQL Server, GridFS, FileTable.*

Постановка проблеми. У кожного користувача персонального комп'ютера під час активного використання програмного забезпечення в розподіленій системі в реальних умовах виникають ситуації, коли з'являються помилки чи фактори, що призводять до збою роботи програми. Через це формуються повідомлення про помилку в програмному забезпеченні, яка може бути відкорегована користувачем. В інших випадках програма сама встигає усунути проблему, і користувач не помічає жодних змін у функціях програми. Незважаючи на те, що проблема вирішена, у таких випадках важливо встановити причину збою й усунути її.

Протягом тривалого часу для розроблення вебдодатків традиційно використовувалися реляційні бази даних із метою зберігання, оброблення й пошуку структурованих даних. У розподілених системах, де працює багато користувачів одночасно, ведеться запис дій, які вони виконують у веб-програмах, у файли типу log. Ці файли надходять до адміністратора в центральному офісі. У них поряд зі штатними діями зафіксовані збої та відмови системи, що трапляються в клієнта. Оперативне оброблення інформації, що знаходиться в log-файлах, дасть змогу вчасно виявити проблеми клієнта та запропонувати шляхи їх усунення. Але з часом log-файлів накопичується все більше й більше, що ускладнює їх аналіз і прийняття рішень щодо усунення проблем, які сталися на боці клієнта. Для вирішення цієї проблеми запропонований і розроблений алгоритм і програмний продукт для оброблення, запису в базу даних log-файлів і їх аналіз із метою діагностування збоїв і відмов програмного забезпечення в розподіленій у мережі Інтернет клієнт-серверній системі.

Аналіз останніх досліджень і публікацій. Порівняльний аналіз реляційної й нереляційної бази даних був проведений С. Тарасовим, який досліджував вставки даних [1, с. 1]. Для тесту був обраний сценарій, що дозволяє таке:

– оцінити придатність СКБД для інтенсивної вставки даних від множини пристроїв;

– оцінити простоту й продуктивність запитів для отриманої таким чином бази даних.

Тест інтенсивної вставки даних від множини датчиків дослідник обмежив 10 млн записів.

На сайті MongoDB була завантажена остання стабільна версія 2.0.2 для 64-розрядної Windows. Альтернативою MongoDB виступав MS SQL Server 2008 R2 Developer Edition, також 64-розрядний. Дані вводились у колекцію MongoDB і таблицю SQL Server, які мають однакову структуру. Відповідні скрипти можна завантажити (MongoDB, SQL Server).

Для SQL-скрипта інтенсивної порядкової вставки в реляційну СКБД додаток спочатку накопичив масив рядків, потім почалася транзакція, відбулася вставка з 10 рядків, що збільшує швидкість вставки в 5 разів порівняно з порядковим внесенням даних.

С. Тарасов відмічає, якщо просто порядково вставляти записи в таблицю, то «нетранзакційність», притаманна MongoDB, виявиться швидше в будь-якому разі. Це свідчить на користь MongoDB, якщо пакетна вставка (BULK INSERT) не завжди прийнятна за логікою програми. Стандартне використання BULK INSERT – масовий

імпорт даних. Власне, для BULK-копіювання немає що тестувати, воно або є в СКБД, або ні. У MongoDB і SQL Server ця функція наявна. Після проведеної вставки й отримання обсягу бази даних помітно, що колекція з 10 млн документів зайняла 3,95 гігабайт, база даних SQL Server – 0,5 Гб (без компресії), тобто швидкість вставки рядків у 8 разів менша, ніж документів.

Хоча час вставки документа приблизно в 3 рази перевищує час вставки рядків у таблицю (пачками по 10), він може бути цілком прийнятним для логіки додатка. Дослідник відмічає невгамовне бажання MongoDB поглинути всю оперативну пам'ять: 4 Гб проти 1 Гб SQL Server при ліміті в 3 Гб. Під час проведення тесту стало зрозуміло, що обсяг використовуваної оперативної пам'яті обмежити не можна. Загалом рішення полягає у створенні виділеного віртуального сервера під СКБД.

Для проведення подальших тестів доводиться робити перезапуск Mongodb для очищення пам'яті. У SQL Server для аналогічного ефекту просто очищаємо буфери й кеш (DBCC). Описані дослідження формують не надто оптимістичний погляд на перспективи використання MongoDB для вирішення поставленої задачі. З еволюцією інтернету й мобільних пристроїв значно зріс обсяг даних, які необхідно зберігати й обробляти. У наш час стає набагато складніше працювати з фіксованими структурами даних. Ще більше складнощів виникає з обробленням неструктурованих даних, особливо якщо потрібна практично необмежена масштабованість.

Постановка завдання. Лог, або логи сервера, лог-файл (словосполучення є синонімами) – це файл текстового формату, у який заносяться дані про всі дії будь-яких користувачів на серверах. Це може бути локальний сервер будь-якої організації чи веб-сервер хост-провайдера, на якому працюють сайти, або FTP-сервера районної мережі [2, с. 1].

У log-файли вноситься докладна інформація про те, який користувач звернувся до ресурсів сервера, його IP-адреса, MAC-адреса його мережевої карти, з якого ресурсу й за якими ключовими запитами був проведений вхід, які сторінки на сервері були відвідані й скільки часу користувач їх переглядав. Крім того, у log-файли записується, які файли були завантажені із сервера чи закачані на сервер. Як ми бачимо, log-файли – це джерело інформації про всіх відвідувачів і користувачів сервера.

Під час кожного звернення будь-якого користувача до веб-сайту відбувається запис про це у файл-журналі. При цьому відбуваються події, які розглянемо детальніше.

1. Здійснення запиту сторінки. Коли користувач вводить в адресному рядку свого браузера ім'я якого-небудь Інтернет-ресурсу й натискає клавішу Enter, браузер здійснює, відповідно до визначених правил, пошук потрібного сервера й передає йому запит на виведення сторінки. У запиті сервера відправляється така інформація:

– IP-адреса комп'ютера, з якого був посланий запит на отримання сторінки (або IP-адреса проксі-сервера, що використовується клієнтом);

– адреса запитуваної Інтернет-сторінки (IP-адреса);

– дата й час відправлення запиту сервера;

– інформація про географічне місцезнаходження клієнта, який здійснив запит до сервера (або місце розташування використовуваного клієнтом проксі-сервера);

– найменування та версія використовуваного клієнтом Інтернет-браузера, з якого й був здійснений запит до сервера;

– адреса тієї веб-сторінки, з якої був здійснений перехід.

2. Передача запитуваної сторінки в браузер клієнта. При цьому сервер здійснює передачу запитуваних даних (які можуть бути представлені у вигляді веб-сторінки, файлу, куки тощо) на комп'ютер клієнта, точніше, у браузер, за допомогою якого була викликана попередня подія.

3. Здійснення запису в log-файл. При цьому сервер здійснює запис усієї отриманої інформації в log-файл, який ще називають журналом подій. Дані в log-файлі вміщують у собі досить незрозумілу інформацію для непідготовленого новачка, але для системних адміністраторів і розробників веб-додатків це цілком зрозумілий і осмислений текст, який є підставою для проведення будь-яких дій або навіть кримінального переслідування клієнта (звичайно, у разі, якщо доступ до ресурсу був неправомірним, наприклад, хакерським зломом, що призвело до виведення з сервера інформації, яка там зберігалася, наприклад бази даних банківських карт клієнтів компанії).

Під час аналізу файлів журналу подій (log-файлів) системним і мережевим адміністраторам слід ураховувати, що отримані дані все ж не є на 100% реальними, відхилення в них зазвичай становить 5–10% через різні технічні й технологічні особливості використовуваного обладнання та його налаштувань. Однак якщо навантаження на сайт є постійним (інакше кажучи, адміністрація

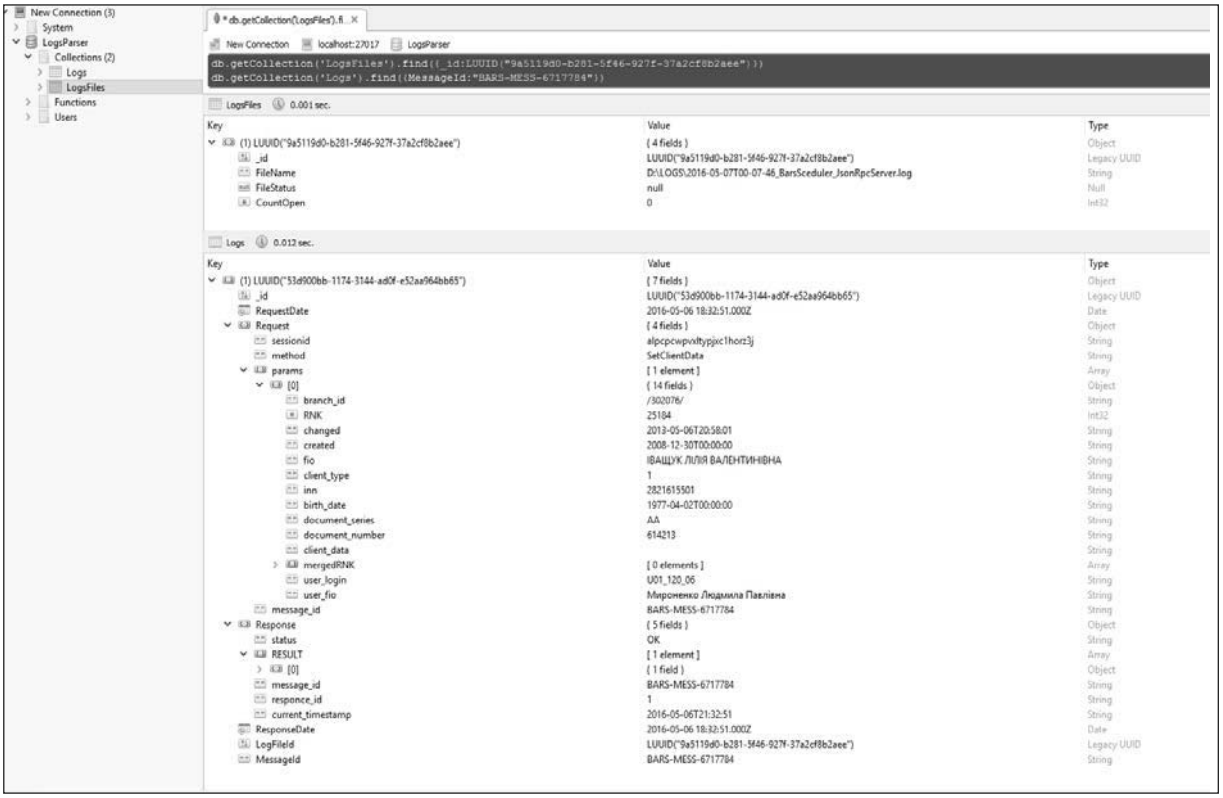

Рис. 1. Інтерфейс RoboMongo

не проводить будь-яких активних рекламних кампаній із залучення відвідувачів, унаслідок чого відвідуваність сайту збільшується на порядки), то отримані під час передачі значення помилки можна вважати більш-менш постійними, що забезпечує можливість здійснення порівняння статистики за минулі періоди.

Звичайно, можна читати й аналізувати й самі логи, але це досить незручний і трудомісткий процес. Оскільки є можливість самому створювати формат логів і їх структуру, то запропоновано зберігати всі збої, зареєстровані в log-файлах системи, у форматі JSON. Для цього було розроблено програмний продукт **LogHelper,** який зберігає log-файли в базі даних MongoDB, переглядає цей файл за заданими критеріями, створюючи документи в колекціях для подальшого аналізу.

Виклад основного матеріалу дослідження. Програмний продукт LogHelper написаний на мові програмування C#, обрана СКБД MongoDB. Алгоритм аналізу для оброблення log-файлів вебсистеми описаний нижче.

На початку аналізу створюється об’єкт Parser(), який матиме як шаблони регулярних виразів, так і два динамічних масиви LogsList, де зберігатимуться об’єкти готових документів для передачі в базу даних, і TempLogsList, де зберігатиметься тимчасовий неповний об’єкт.

Далі відкривається файл і передається в метод самого аналізу й вибірки даних. На початку методу створюються дві змінних dataStart, dataEnd – дати запиту й відповіді в одному з рядків, знайдених у файлі, value – сам рядок. Потім відтворюється цикл, який перебирає всі рядки файлу. У циклі створюються дві змінні, де записується дата запиту й відповіді. Після знайденої дати відбувається пошук за шаблоном Input, Output, а після знайденого збігу запускається метод CreateLog(), у який передаються messageId, requestDate, request, responseDate, response, logFile. У методі створюється змінна log, де записується перший знайдений документ у тимчасовому масиві var log = TempLogList. FirstOrDefault(o=>o.id==id). Якщо в результаті буде null, тоді відбуватиметься створення нового log-документа, що додається в тимчасовий масив із неповними даними із запитом або відповіддю. Але якщо об’єкт буде знайдений у тимчасовому масиві, тоді відбуватиметься перевірка вхідних даних. Тобто якщо request ! = ””, тоді в знайдений об’єкт у тимчасовому масиві буде переданий requestDate і request, або навпаки, responseDate, response. Після успішного створення

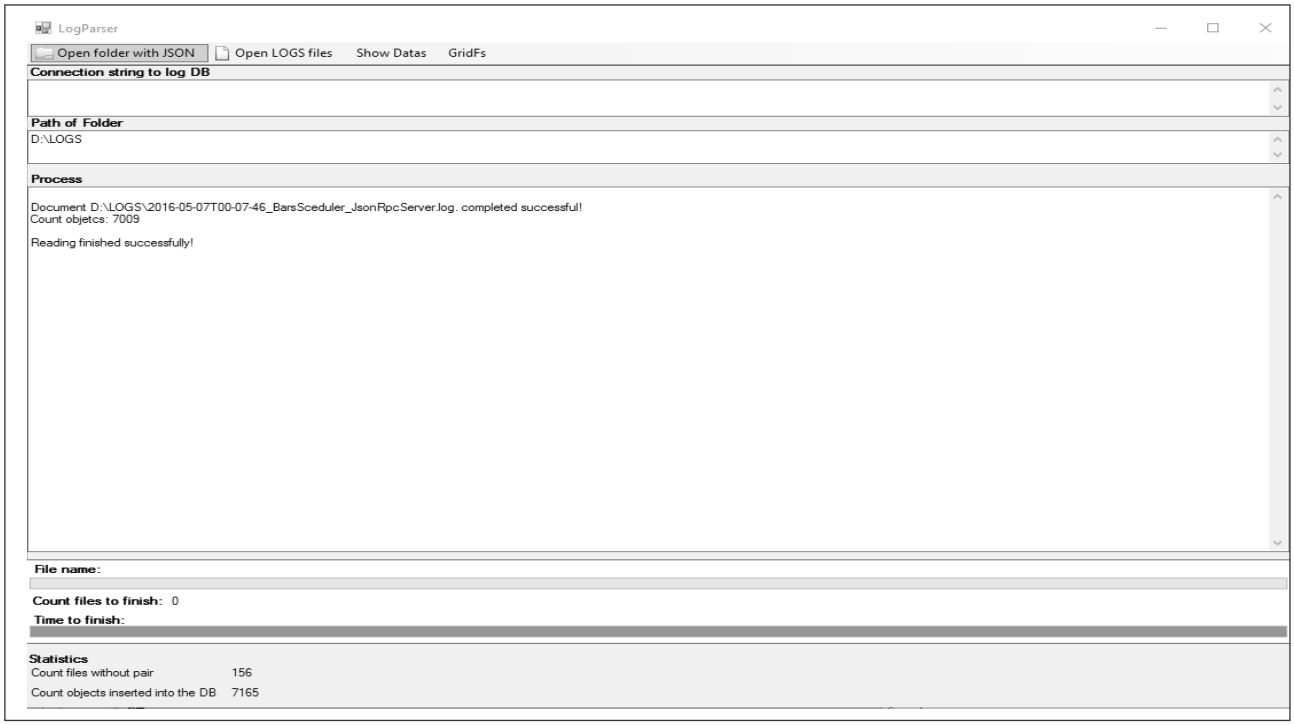

Рис. 2. Інтерфейс LogHelper, аналіз log-файлі

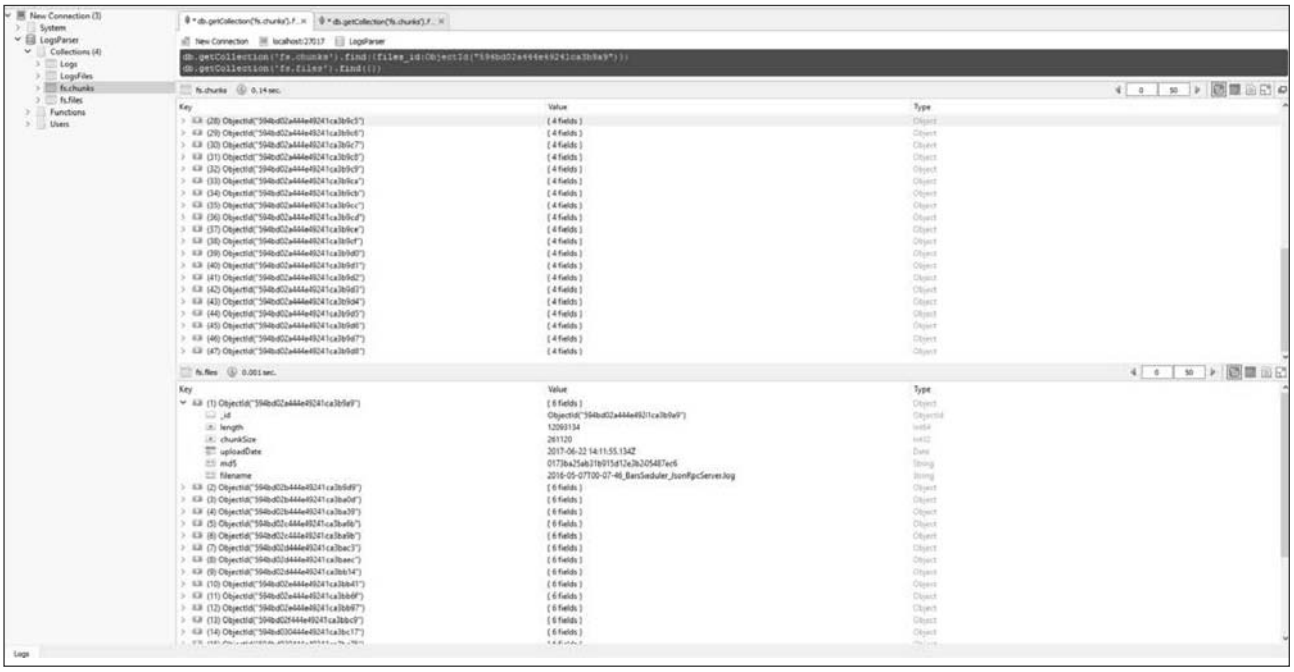

Рис. 3. Інтерфейс завантажених проаналізованих log-файлами відомостей у базу даних MongoDb

чи доповнення об'єкта відбуватиметься перевірка на повноту. Якщо MessageId ! = "" && log.Request ! = "" && log.Response ! = ""), тоді додаємо готовий, повний об'єкт LogsList.Add(log) і видаляємо з тимчасового масиву TempLogsList.Remove(log). І так – доти, доки не переберемо всі рядки файлу. Після завершення перебору записуємо всі повні документи в базу даних MongoDb і перевіряємо, чи щось залишилося в тимчасовому масиві. Якщо так, тоді заповнюємо об'єкт шаблоном «Дані не знайдено й стандартне налаштування дати».

Розглянемо роботу алгоритму на прикладі невеличкого фрагмента з log-файлу, в якому зберігаються всі запити й відповіді від сервера.

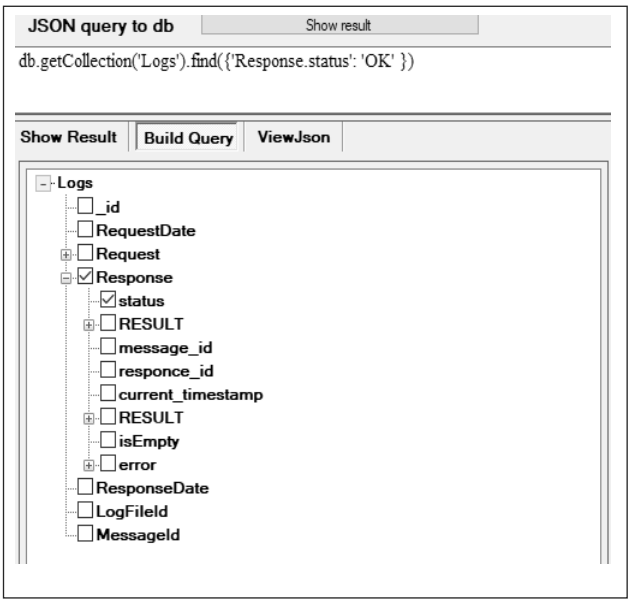

Рис. 4. Інтерфейс побудови запиту в LogHelpe

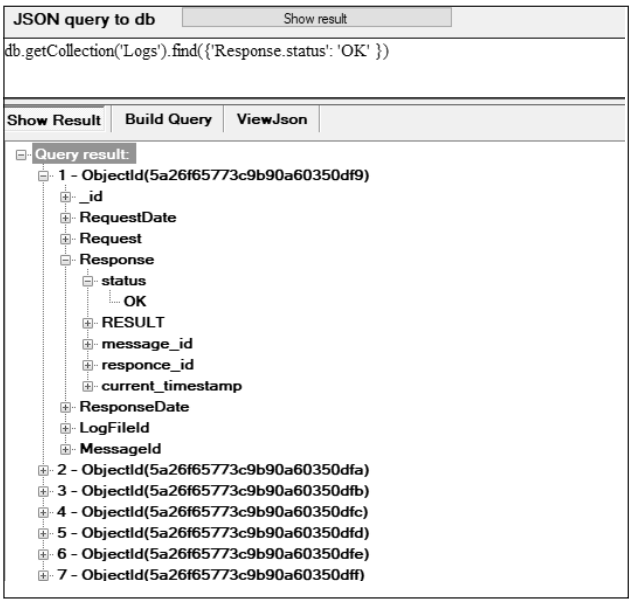

Рис. 5. Інтерфейс результату запиту в LogHelper

=> BarsScheduler at 5/6/2016 9:32:51 PM : from 10.7.73.12 ; RequestType = POST; ContentType = application/json; charset="UTF-8"; Input = {"sessionid":"alpcpc wpvxltypjxc1horz3j","method":"SetClientData", "params":[{"branch_id":"/302076/","RNK":25184," changed":"2013-05- 06T20:58:01","created":"2008-12-30T00:00:00","fio":"ІВАЩУК ЛІЛІЯ ВАЛЕНТИНІВНА","client_type":"1","inn": "2821615501","birth_date":"1977-04-02T00: 0:00","document_series":"AA","document_ number":"614213","client_ data":"", "mergedRNK":[],"user_ login":"U01_120_06","user_fio":"Мироненко Людмила Павлівна"}],"message_id":"BARS-MESS-6717784"}

<= BarsScheduler at 5/6/2016 9:32:51 PM : to 10.7.73.12 Output = {"status":"OK","RESULT":[{" RNK":"25184"}],"message_id": "BARS-ME

Наведений приклад показує завантаження файлу в MongoDb. Загальний розмір – 43, файли займають 504 Мб. Вставка тривала 10,05 секунди. Розмір колекції з файлами – 265 Мб. Як можна побачити, MongoDb стискає колекцію майже вдвічі, швидкість при цьому прийнятна.

На рис. 3 можна побачити, шо один файл розділений на 47 фрагментів однакового розміру. У кожному фрагменті, або шарді, є id (ідентифікатор) файлу й бінарні дані документа. Після занесення log-файлу до БД у MongoDb можна легко побудувати JSON запити й проводити аналіз даних, наприклад, за такими параметрами, як дата, місце розташування, статус виконання тощо.

За наведеним алгоритмом за секунду обробляється 1400 рядків і створюється 344 повних об'єкти, які готові для збереження в базі даних. Додатково програмний продукт здатний не тільки аналізувати файл і записувати повні об'єкти в базу даних, а й проводити аналіз наявних даних у базі й запропонувати можливі варіанти створення запитів для користувача (йому залишається тільки ввести дані, за якими буде працювати запит для вибірки даних із бази даних, де збережені проаналізовані log-файли).

За допомогою запиту користувач зможе легко прочитати лог-об'єкт у зручному вигляді дерева й провести аналіз за результатами запиту.

У цій програмі є ще одна велика й важлива відмінність. Під час аналізу log-файлів виділяються й зберігаються об'єкти, які мають статус error, в окрему таблицю, а також інформація про помилку (код помилки, повідомлення тощо). Потім за допомогою цих даних із легкістю можна скласти запит і виявити причину відмови чи збою. Згодом, проаналізувавши дані про помилки, у програмі можна передбачити можливість створення таблиці з кодами помилок і їх рішенням, щоб програма могла самостійно проаналізувати та запропонувати варіанти вирішення виявленої проблеми для користувача за прийнятими log-файлами.

Висновки. У результаті проведеного дослідження з'ясовано, що log-файли є вельми потужним засобом для аналізу функціонування клієнтських додатків розподіленої системи на сервері. Використання LogHelper дає змогу побудувати чіткі й точні запити для аналізу log-файлів і за допомогою

зручної та швидкої технології GridFs зберігати будь-які файли в базі даних у фрагментованому вигляді, якщо порівняти з реляційною базою даних.

За допомогою підказки побудови фільтрів є можливість зібрати статистику та структурування отриманих даних на основі ключових слів для формування підсумкового документа про збої, що відбулися в програмному забезпеченні клієнта за заданий період. Програмний продукт у складі системи підтримки прийняття рішень пропонується для використання аналітиками та програмістами. Із його допомогою можна визначити причину відмови, побудувавши запит за певними параметрами. Згодом планується створення бази знань експертів, яка буде містити шаблони несправностей і варіанти їх виправлення. Усі ці заходи сприятимуть підвищенню надійності функціонування розподіленої системи.

Список літератури:

1. Возможности MongoDB. URL: http://habrahabr.ru/post/119703/.
2. Логи: разбираемся с понятием лог-файл. URL: http://thelocalhost.ru/log/.

ОБРАБОТКА И АНАЛИЗ LOG-ФАЙЛОВ С ПОМОЩЬЮ ПРОГРАММНОГО ПРОДУКТА LOGHELPER

В статье рассмотрен программный продукт, который работает с log-файлами, непонятными для простого пользователя. Предложен метод сохранения log-файлов, их анализа и принятия решений по исправлениям обнаруженных ошибок пользователя в распределенной системе. Описана структура log-файлов, выяснено их назначение. Рассмотрен формат JSON и доказано, почему именно в этом формате целесообразно описывать логи. Также определен алгоритм анализа и сохранения файлов с помощью GridFS в MongoDB. После обработки данных предложено решение выявленных отказов в системе. С помощью программного продукта проведен сравнительный анализ параметров хранения и обработки log-файлов в реляционных (SQLServer) и нереляционных (MongoDB) базах данных.

Ключевые слова: *базы данных, SQL, NoSQL, JSON, реляционные СУБД, нереляционные СУБД, MongoDB, SQL Server, GridFS, FileTable.*

LOG FILES PROCESSING AND ANALYSIS BY LOGHELPER SOFTWARE PRODUCT

The article discusses a software product that works with log files that are unclear to the simple user. The method of saving log files, their analysis and making decisions on corrections of detected errors of the user in the distributed system is offered. The structure of log-files is described, their purpose is clarified. The format of JSON is considered and it is proved why in this format it is expedient to describe the logos. An algorithm for analyzing and saving files using GridFS in MongoDB is also defined. After the data processing, the solution of the revealed failures in the system is proposed. Using the software product, a comparative analysis of the parameters for storing and processing log files in relational (SQLServer) and non-relational (MongoDB) databases was conducted.

Key words: *database, SQL, NoSQL, JSON, BSON, relational database, non-relational database, MongoDB, SQL Server, GridFS, FileTable.*