УДК 04.65

RESEARCH AND DEVELOPMENT OF THE METHOD OF PROCESSING LOG FILES IN A DISTRIBUTED INFORMATION SYSTEM USING NON-RELATIONAL DATABASE MONGODB

V. Bratskyi
National University of Food Technologies

Key words:
Database
SQL
NoSQL
JSON
BSON
Relational database
Non-relational database
MongoDB
SQL Server
GridFS
FileTable

Article history:
Received 11.01.2018
Received in revised form 26.01.2018
Accepted 06.02.2018

Corresponding author:
V. Bratskyi
E-mail:
vadymbratskyi@gmail.com

ABSTRACT

Every user of a personal computer, when actively using software in a distributed system, in real conditions, there are situations when there are errors or factors that cause program to fail. In such situations, an error message is generated in the software, which can be corrected by the user. In other cases, the program itself has time to fix the problem, and the user does not notice any changes in the functions of the program. In spite of the fact that the problem is solved, in such cases, it is important to establish the cause of the failure and eliminate it.

This article deals with this topic, where we will get acquainted with such moments where the software itself bypasses the errors and fixes them in log files that are not clear to the average user. The solution to save log files, parsing them, analyzing and making decisions about fixing errors found on the user's side in the distributed system is also considered. The structure of the log-files is described and their purpose is inquired. JSON format is considered and it is proved why it is advisable to describe logos exactly in this format. It is also considered the parsing and file saving algorithm using GridFS in MongoDB. After the data processing, suggestions were made for solving the discovered failures in the system. And also with the help of the software product, a comparative analysis of parameters for storing and processing log files in the relational (SQLServer) and non-relational (MongoDB) databases is conducted.

Therefore, this topic is currently relevant to each developer and system analyst, which simultaneously runs a large number of users. For developers, this analysis will help to diagnose software errors, to move in the right direction, and for analytics — to design a project to avoid these errors in the future.

DOI: 10.24263/2225-2924-2018-24-1-4

ДОСЛІДЖЕННЯ ТА РОЗРОБКА МЕТОДУ ОБРОБКИ LOG-ФАЙЛІВ У РОЗПОДІЛЕНІЙ ІНФОРМАЦІЙНІЙ СИСТЕМІ З ВИКОРИСТАННЯМ НЕРЕЛЯЦІЙНОЇ БАЗИ ДАНИХ MONGODB

В.О. Брацький
Національний університет харчових технологій

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

У статті розглянуто програмний продукт, що сам обходить помилки і фіксує їх у log-файлах, які є незрозумілими для простого користувача. Запропоновано метод збереження log-файлів, їх парсинг-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 парсингу і збереження файлів за допомогою GridFS в MongoDB. Після обробки даних запропоновано вирішення виявлених відмов у системі. За допомогою програмного продукту проведено порівняльний аналіз параметрів збереження та обробки log-файлів у реляційній (SQLServer) і нереляційній (MongoDB) базах даних.

Тема, що розглядається, є актуальною для кожного розробника й аналітика систем, де одночасно працює велика кількість користувачів. Проведений аналіз полегшить діагностування помилок програмного забезпечення розробникам, допоможе рухатися в правильному напрямку, дасть змогу аналітикам розробити проект так, щоб ці помилки в подальшому не проявлялися.

Ключові слова: *бази даних, SQL, NoSQL, JSON, BSON, реляційні СКБД, нереляційні СКБД, MongoDB, SQL Server, GridFS, FileTable.*

Постановка проблеми. Протягом тривалого часу для розробки веб-додатків традиційно використовувалися реляційні бази даних з метою зберігання, обробки і пошуку структурованих даних. У розподілених системах, де працює водночас багато користувачів, ведеться запис дій, які вони виконують у веб-програмах, у файли типу log. Ці файли надходять до адміністратора в центральному офісі. В них, поряд зі штатними діями, зафіксовано збої та відмови системи, що трапляються у клієнта. Оперативна обробка інформації, що знаходиться в log-файлах, дасть змогу вчасно виявити проблеми клієнта та запропонувати шляхи їх усунення. Але з часом log–файлів накопичуються

все більше і більше, що ускладнює їх аналіз і прийняття рішень щодо усунення проблем, які сталися на боці клієнта. Для вирішення цієї проблеми запропоновано і розроблено алгоритм та програмний продукт для обробки, запису в базу даних log-файлів та їх аналіз з метою діагностування збоїв і відмов програмного забезпечення в розподіленій в мережі Інтернет, клієнт-серверній системі.

Метою дослідження є розробка та застосування алгоритму і програми обробки даних з log-файлів для виявлення збоїв і відмов в розподіленій системі, а також формування рішень щодо їх усунення.

Викладення основних результатів дослідження. *Призначення і структура log-файла.* Лог, або логи сервера, лог-файл (словосполучення є синонімами) — це файл текстового формату, в який заносяться дані про всі дії будь-яких користувачів на серверах. Це може бути локальний сервер будь-якої організації або веб-сервер хост-провайдера, на якому працюють сайти, або FTP-сервера районної мережі [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 і за якими ключовими запитами був проведений вхід, які сторінки на сервері були відвідані і скільки часу він їх переглядав. Крім того, в log-файли записується, які файли були завантажені із сервера або закачані на сервер. Як ми бачимо, log-файли — це джерело інформації про всіх відвідувачів і користувачів сервера.

При кожному зверненні 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 і натискає клавішу Enter, браузер здійснює, відповідно до визначених правил, пошук потрібного сервера і 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 і час відправки 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 і був здійснений запит до сервера;
- адреса тієї веб-сторінки, з якої і 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 в браузер, за допомогою якого було викликано попередню подію.

3. Здійснення запису в log-файл. При цьому сервер здійснює запис усієї отриманої інформації в log-файл, який ще називають журналом подій. Дані у

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, в разі, якщо доступ до ресурсу був неправомірним, наприклад, хакерським зломом, що призвело до виведення з сервера інформації, яка там зберігалася, бази даних банківських карт клієнтів компанії).

Але при аналізі файлів журналу подій (log-файлів) системним і мережевим адміністраторам слід враховувати, що отримані дані все ж не є на 100% реальними, відхилення в них зазвичай становить 5—10% через різні технічні і технологічні особливості використовуваного обладнання та його налаштувань. Однак, якщо навантаження на сайт є постійним (інакше кажучи, адміністрація не проводить будь-яких активних рекламних кампаній із залучення відвідувачів, внаслідок чого відвідуваність сайту збільшується на порядки), то отримані при передачі значення помилок можна вважати більш-менш постійними, що забезпечує можливість здійснення порівняння статистики за минулі періоди.

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

Формат JSON. JSON (англ. JavaScript Object Notation) — текстовий формат обміну даними, що базується на JavaScript. Як і багато інших, текстові формати JSON легко читаються. Формат JSON був розроблений Дугласом Крокфордом [2].

Незважаючи на походження від JavaScript (точніше, від підмножини мови стандарту ECMA-262, 1999 р.), формат вважається незалежним від мови і може використовуватися практично будь-якою мовою програмування. Для багатьох мов існує готовий код для створення й обробки даних у форматі JSON.

За рахунок своєї лаконічності, якщо порівняти з XML, формат JSON більше підходить для серіалізації складних структур. Якщо говорити про веб-додатки, він доречний у задачах обміну даними як між браузером і сервером (AJAX), так і між самими серверами (програмні HTTP-сполучення).

Наступний приклад показує JSON-представлення об’єкта, що описує людину. В об’єкті є строкові поля імені і прізвища, об’єкт, що описує адреси, і масив, що містить список телефонів. Як видно з прикладу, значення може являти собою вкладену структуру.

{“FirstName”: “Іван”,
“LastName”: “Іванов”,
“Address”: {“StreetAddress”: “пр. Науки, 26, кв. 5—4”,
“City”: “Київ”,
“PostalCode”: 000101},
“PhoneNumbers”: [“812 123-1234”, “916 123-4567”]}.

Якщо порівняти з мовами XML або SQL, то можна помітити, що JSON має більш компактний вигляд і краще читається.

Парсер log-файлів. Розроблений парсер і алгоритм обробки даних призначені для швидкого і лаконічного аналізу великої кількості log-файлів. Файли можна як завантажувати в базу даних (БД), так і напряму звертатися до парсера. Для завантаження файлу в базу даних використовується методика GridFS [3]. Це специфікація MongoDB для зберігання великих файлів. Тобто ця технологія розбиває великі файли на менші фрагменти. Фрагменти зберігаються в одній колекції (fs.chunks), а метадані про фото або текстовий документ — в іншій колекції (fs.files). Коли користувач робить запит до файлу, то GridFS звертається до колекції з фрагментами. Кожен фрагмент має id (ідентифікатор) файлу і за id GridFS збирає файл повністю та повертає користувачеві.

Кілька переваг GridFS:

- GridFS сам виконує реплікацію або сегментування (шардінг);
- MongoDB розділяє файли на фрагменти по 2 Гб, тому в ОС не виникатимуть проблеми з маніпулюванням файлами;
- не виникає потреба обмежувати ОС на імена файлів або кількість файлів в одній директорії;
- MongoDB автоматично генерує і зберігає MD5 хеш збережуваного файлу. Це зручно для порівняння завантажених файлів по MD5 хешу і виявлення дублікатів або валідації успішного завантаження.

Програмний продукт парсера написано на мові програмування C#, обрана СКБД MongoDB. Алгоритм парсера для обробки log-файлів веб-системи описаний нижче.

На початку парсингу 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 парсингу і вибірки даних. На початку методу створюються дві змінних *dataStart, dataEnd* — дати запиту і відповіді в одному з рядків знайдених у файлі, *value* — сам рядок. Потім відтворюється цикл, який перебирає всі рядки файлу. В циклі створюються дві змінні, де записуються дата запиту і відповіді. Після знайденої дати відбувається пошук за шаблоном *Input*, *Output*, а після знайденого бігу запускається метод *CreateLog()*, в 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 з неповними даними із запитом або відповіддю. Але якщо об'єкт буде знайдений у тимчасовому масиві, тоді відбуватиметься перевірка вхідних даних. Тобто якщо *reques != ""*, тоді в знайдений об'єкт у тимчасовому масиві буде передані *requestDate* і *request*, або, навпаки, *responseDate*, *response.* Після успішного створення або доповнення об'єкта відбуватиметься

перевірка на повність. Якщо *MessageId != “” && log.Request != “” && log.Response != “”),* тоді додаємо готовий, повний об’єкт *LogsList.Add(log)* і видаляємо з тимчасово масиву *TempLogsList.Remove(log)*. І так до тих пір, 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 і стандартне налаштування дати*».

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

=> BarsScheduler at 5/6/2016 9:32:51 PM : from 10.7.73.12 ; RequestType = POST; ContentType = application/json; charset=“UTF-8”;

Input = {“sessionid”:“alpcpcwpvxltypjxc1horz3j”,“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”:“1977-04-02T00:00:00”,“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-MESS-6717784”,“responce_id”:“1”,“current_timestamp”:“2016-05-06T21:32:51”}.

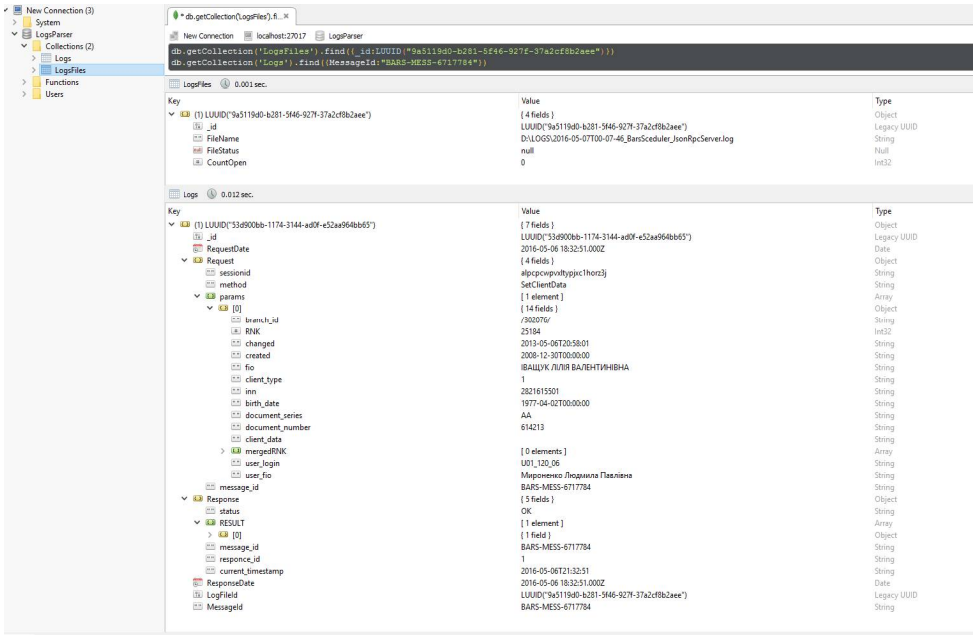

Рис. 1. Інтерфейс RoboMongo

Інтерфейс RoboMongo наведено на рис. 1, інтерфейс парсера — на рис. 2.

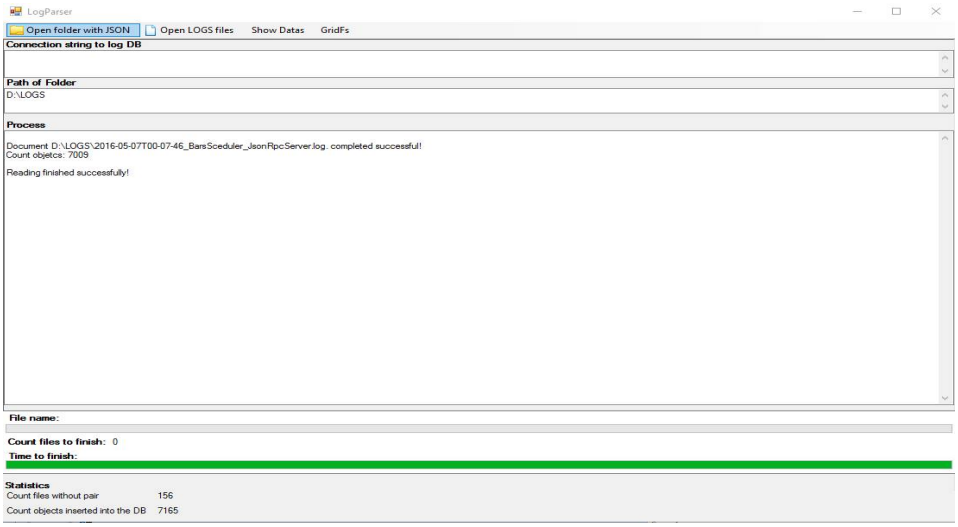

Рис. 2. Інтерфейс парсера log-файлів

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

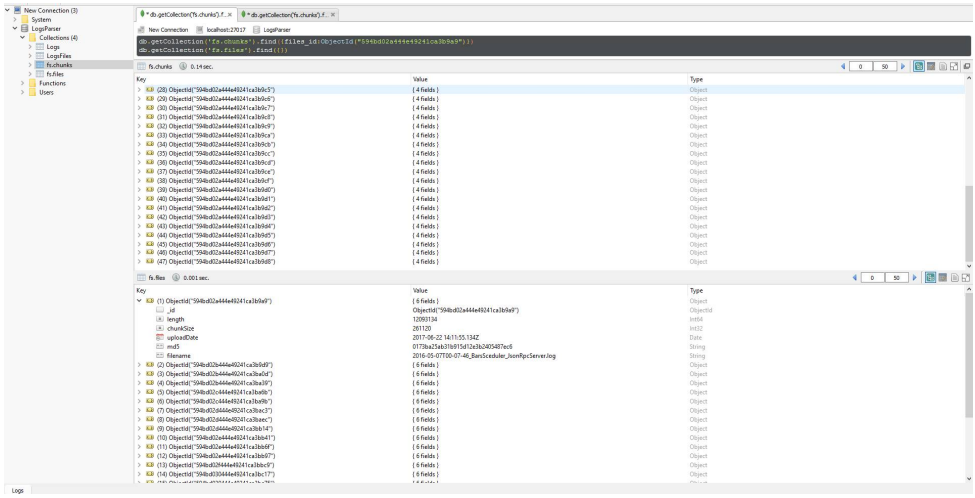

Рис. 3. Інтерфейс бази даних з log-файлами

З рис. 3 можна побачити, шо один файл розділено на 47 фрагментів рівномірного розміру. В кожному фрагменті, або шарді, є id (ідентифікатор) файлу і бінарні дані документа.

Після занесення log-файлу до БД у MongoDb, можна легко побудувати JSON запити і проводити аналіз даних, наприклад, за такими параметрами, як дата, місце розташування, статус виконання тощо.

Порівняння технології збереження log-файлів у SQL Server і MongoDb. Традиційним способом зберігання в БД неструктурованих двійкових файлів, таких як відео, аудіо, картинки тощо, є визначення в таблиці поля типу varbinary (max) або застосування FileStream (з'явилося в SQL Server 2008). У у SQL Server 2012 з'явився новий підхід з використанням таблиць FileTable. Технологія FileTable базується на функціоналі FileStream, розширюючи при цьому його можливості. У FileTable для організації структури каталогів використовується тип HierarchyId [4].

Для створення таблиці, в якій будуть зберігатися файли, потрібно провести незначні модифікації з базою даних і сервером, як описано в документі [5]. В результаті створюється директорія, яку слід задати в налаштуваннях сервера і базах даних. Саме там будуть зберігатися всі дані FileStream. У SQL Server Management Studio в розділі Tables → File tables буде відображено таблицю з log-файлами. Для додавання файлів у задану директорію можна використовувати як систему CRUD, так і провідники, які автоматично будуть завантажувати log-файли у базу даних.

І тут же ми можемо помітити, що при збережені файлів у MS SQL Server розмір бази даних буде залежати від кількості файлів і їх розміру. А у файловій системі у MongoDb колекція з файлами стиснена майже вдвічі і база даних є одним документом, який легко транспортувати.

Особливість FileTables полягає в тому, що ця технологія об'єднує компонент SQL Server Database Engine з файлової системою NTFS, розміщуючи дані великих двійкових об'єктів (BLOB) типу varbinary (max) у файловій системі. Виходячи з цього, можна виділити такі переваги:

- можливий нетранзакційний доступ через файлову систему Windows, при цьому продуктивність дорівнює продуктивності файлової системи;
- для кешування файлів у сховище FileTable використовується системний кеш NT, при цьому SQL-буфер використовується тільки для обробки запитів;
- розмір виконуваного файлу обмежений тільки розміром NTFS-розділу;
- можливий також транзакційний доступ за допомогою звичайних інструкцій SELECT, INSERT, UPDATE і DELETE;
- доступне використання інтегрованих служб SQL Server, таких як резервне копіювання, вбудована підтримка безпеки, повнотекстовий пошук тощо.

Під час розробки програми проводилося порівняння SQL Server 2014 і MongoDB. Програма розроблена з асинхронними можливостями і проблемою у MongoDB було збереження оброблених даних у базі даних. Якщо під час збереження даних відбувалася обробка іншого файлу, в іншому потоці, то його збереження переривало попереднє і всі попередні документи не потрапляли до бази даних. У БД SQL Server в цьому випадку все нормально — сам сервер виставляє потоки в чергу і не переривае дії попередніх. Тому було прийняте рішення у БД MongoDb для кожного потоку створювати окреме з'єднання з базою даних, і кожне введення даних з парсера має працювати в окремому потоці в окремому MongoClient. Це є одним із мінусів MongoDb порівняно з SQL Server.

Перевагою БД у MongoDb є його зручність і читабельність log-файлів у форматі JSON, з яким база даних легко взаємодіє. У MongoDb швидше вико-

нуються запити для аналізу даних, ніж у БД в SQL Server, де пошук відбувається послідовно по рядках таблиці для виявлення даних, подібних до параметра пошуку.

Висновок

У результаті проведеного дослідження з'ясовано, що log-файли є вельми потужним засобом для аналізу функціонування клієнтських додатків розподіленої системи на сервері. Використання MongoDb дає змогу побудувати чіткі і точні запити для аналізу log-файлів і за допомогою зручної та швидкої технології GridFs зберігати будь-які файли у базі даних у фрагментованому вигляді, якщо порівняти з реляційною базою даних [6]. Для збору статистики та структурування отриманих даних планується реалізація фільтру на основі ключових слів для формування результуючого документа про збої, що відбулися у програмному забезпеченню клієнта за заданий період.

Програмний продукт у складі системи підтримки прийняття рішень пропонується для використання аналітиками та програмістами. З його допомогою можна визначити причину відмови, побудувавши запит за певними параметрами. В подальшому планується створення бази знань експертів, яка буде містити шаблони несправностей і варіанти їх виправлення. Всі ці заходи сприятимуть підвищенню надійності функціонування розподіленої системи.

Література

1. Логи: розбираємося з поняттям лог-файл [Електронний ресурс]. — Режим доступу : http://thelocalhost.ru/log/.

2. Формат JSON [Електронний ресурс]. — Режим доступу : https://learn.javascript.ru/json.

3. Symfony — загрузка файлів у MongoDB GridFS [Електронний ресурс]. — Режим доступу : https://habrahabr.ru/post/314840/.

4. Зберігання ієрархічних структур даних в Microsoft SQL Server [Електронний ресурс]. — Режим доступу : http://ts-soft.ru/blog/hierarchyid.

5. Таблиці файлів в SQL Server 2012 [Електронний ресурс]. — Режим доступу : http://ts-soft.ru/blog/filetable.

6. *Брацький В.О.* Дослідження особливостей застосування реляційних і нереляційних баз даних на прикладі SQL Server та MongoDb / В.О. Брацький, О.М. М'якшило // Наукові праці Національно університету харчових технологій. — 2016. — Том 22, № 5. — С. 15—23.