ЕЛЕКТРОННО СПИСАНИЕ „ИКОНОМИКА И КОМПЮТЪРНИ НАУКИ“, БРОЙ 2, 2019,
ISSN 2367-7791, ВАРНА, БЪЛГАРИЯ
ELECTRONIC JOURNAL “ECONOMICS AND COMPUTER SCIENCE”, ISSUE 2, 2019,
ISSN 2367-7791, VARNA, BULGARIA

Специфични административни задачи при MongoDB и MySQL

Иван КУЮМДЖИЕВ[1]

[1] Икономически университет - Варна
ivan_ognyanov@ue-varna.bg

Резюме. Системите за управление на бази от данни определят до голяма част от ограниченията, които ще имат използващите ги софтуерни приложения. По тази причина администраторите на бази от данни се грижат за управление на редица процеси от които зависи качеството на софтуера. Публикацията има за цел да бъдат изследвани и сравнени методите за извършване на основни административни задачи при извършване на архивиране и възстановяване на бази от данни. За обект на сравнението са избрани системите за управление на бази от данни MySQL и MongoDB. Анализирани са приликите и разликите в категориите логическа и физическа организация на данните, методи за измерване размера на базата от данни, нарастването ѝ, както и времето за изпълнение на операции по архивиране и възстановяване. Изследвани са само безплатните версии на продуктите в качеството им на най-разпространени в световен мащаб. Открити са някои сходства в управлението на двете системи, както и силни и слаби страни на всяка от тях спрямо другата. Изследването може да бъде полезно за администратори на бази от данни

Ключови думи: администриране на бази от данни, архивиране, сравнение на бази от данни, системи за управление на бази от данни.

1. Въведение

Познаването на характеристиките и методите за измерване за количествено измерване на качествата на СУБД е важна част от отговорностите на администраторите им. Това може да помогне за подобряване на бързодействието, повишаване на сигурността и създаване на устойчива стратегия за архивиране и възстановяване.

Публикацията разглежда безплатните версии на две от най-популярните СУБД в световен мащаб – MySQL и MongoDB. Причините за широкото им разпространение могат да бъдат свързани с възможността за безплатно използване и скорост на работа и при двете; отворен код, използване на разпределена обработка на информацията и нерелационна схема в MongoDB. Целта на публикацията е да открои методи за извършване на следните административни задачи - измерване размера на базата от данни и откриване на тенденции в нейния растеж, методите за измерване на времето необходимо за изпълнение на команди за архивиране и възстановяване, както и да подчертае силни и слаби страни на двете СУБД в тези области.

2. Особености при логическа и физическа организация на данните в MySQL и MongoDB

Размерът на базата от данни е фактор, оказващ влияние, както на процеса на архивиране, така и на процеса на възстановяване. За да може да бъде изследвана степента му на значимост, е необходимо да се изследват характеристиките на MySQL, оказващи влияние на размера на базата от данни, методите за измерване на размера, както и добиване на информация, позволяваща увеличаването му със запазване на тенденциите за растеж.

Архитектурата на MySQL позволява избор на различни хранилища за всяка от таблиците в базата от данни. (Bell, 2013) определя използването на различни хранилища в MySQL като една от най-добрите възможности предлагани от продукта, защото това позволява да бъде настроена базата данни като се избира най-подходящото хранилище в зависимост от изискванията на приложенията към всяка таблица. Например използване на хранилище, което осигурява контрол на транзакциите за много активни бази от данни или използване на хранилище, което съхранява данните в оперативната памет в случаи, в които те се достъпват често, но не се променят.

MySQL 5.7 поддържа следните хранилища за данни: MyISAM, MERGE, MEMORY, InnoDB, CSV, Blackhole, Archive, Example (шаблон за създаване на нови хранилища) и FEDERATED (MySQL, 2019g), като представя всяка таблица на твърдия диск като поне един файл. Той има разширение .frm, съдържа описание на структурата на таблицата и се създава от сървъра, а различните хранилища могат да създадат допълнителни файлове, съдържащи данните от таблиците и информация за индексите, като имената и структурата им зависи от хранилището (DuBois, 2008).

MyISAM е типът хранилище на данни по подразбиране в MySQL до версия 5.5. Таблиците са създадени чрез методи за компресия и оптимизация на индексирането за подобряване на скоростта и по тази причина се използва при реализация на складове от данни и приложения за електронна търговия (Bell, 2013), както и при статистически анализ на данните (Kaneva, 2019)(Vasilev and KEHAYOVA-STOYCHEVA, 2017). Заключването е на ниво таблица и изборът на това хранилище се препоръчва, когато скоростта на четене е от най-голямо значение. MyISAM представя всяка таблица като три файла на твърдия диск – освен файла, описващ структурата на таблицата, се използват и файлове с разширение .myd (за запис на съдържанието на редовете) и .myi (запис на информация за индексите) (MySQL, 2019g).

InnoDB поддържа механизъм за превъртане назад на неизпълнени до край транзакции, като таблиците от този вид могат да нарастват до 64 TB (по-малко от MyISAM – 256TB). Въпреки по-малкия максимален размер хранилището се използва по подразбиране за всички новосъздадени таблици и от фирмата разработчик (Oracle) препоръчват прилагането му. InnoDB е първото хранилище в MySQL, което поддържа работа с транзакции (съгласуване с принципа ACID) и поддържане на референтна цялост на данните. Таблици, съхранявани в такъв формат, предлагат заключване на ниво запис (единствено платеното хранилище NDB предлага подобна възможност) – докато се изпълнява транзакция върху дадени записи от таблицата, те остават заключени, но други транзакции могат да се изпълняват върху останалите записи.

Таблиците от тип InnoDB използват два начина за съхраняване на данните: индивидуално и споделено таблично пространство. Споделеното се състои от един или повече големи файлове, които формират логически свързана област за съхраняване на данни с размер равен на сумата от размерите на индивидуалните файлове (DuBois, 2008). В такъв случай единственият файл специфичен за всяка таблица е файлът с разширение .frm. При използване на опциите, зададени по подразбиране в MySQL (до версия 5.6), се създава единствен файл с данни с име ibdata1 (нарича се споделено таблично пространство) и два лог файла ib_logfile0 и ib_logfile1. В тези файлове се записват данните от таблиците от всички бази от данни на сървъра, както и операциите, извършвани върху тях. След версия 5.6 по подразбиране (MySQL, 2019b) е включена опцията innodb_file_per_table, при която за всяка таблица се създава файл с данни с разширение .ibd.

Хранилището позволява създаване и изтриване на индекси с по-малко влияние на производителността и достъпността, а структурата му е ефективна за полета с дълъг текст и BLOB. Според нас това е и една от причините да се препоръчва за среди, в които се извършват множество операции по запис, докато MyISAM е препоръчително за таблици предимно за четене.

Хранилищата Merge и Federated имат сходно предназначение. Докато първото е виртуална таблица създадена, чрез комбиниране на няколко таблици с еднаква структура от тип MyISAM от един сървър (Vaswani, 2009), то Federated може да се базира на таблици от различни сървъри (Bell, 2013).

Таблиците от тип Memory използват единствено файл с разширение .frm и не се представят по друг начин във файловата система, защото данните и индексите й се записват в оперативната памет (MySQL, 2019f). При спиране на сървъра съдържанието се изтрива и при рестартиране таблиците са празни. Таблиците от тип Blackhole също се представят на твърдия диск с единствен файл с разширение .frm, но за разлика от Memory таблиците не съдържат никакви данни (MySQL, 2019d).

Хранилището Archive е създадено за съхраняване на големи количества рядкодостъпвани исторически данни в компресиран вид. Няма индексация и единственият начин за достъп до данните е чрез сканиране на таблицата (MySQL, 2019c).

Описаните характеристики на хранилищата в MySQL но видят до извода, че от гледна точка на архивирането и възстановяването, значение имат предимно хранилищата MyISAM и InnoDB – използват се най-често и за разлика от Memory и Blackhole съхраняват оперативни данни на твърдия диск.

MongoDB е значително по-нов софтуерен. Въпреки това добива голяма популярност и се използва от широк кръг световноизвестни компании за обслужване на части от информационните им системи, поддържащи големи обеми от данни. Нарастващата популярност на MongoDB налага нуждата да бъдат оценени възможностите й за изпълнение на класически административни задачи.

MongoDB е СУБД, създадена за бърза разработка на уеб приложения (Vohra, 2015), и по тази причина е проектирана да дава по-голяма свобода на програмистите при проектиране на базата от данни. За разлика от системите за управление на релационни бази от данни като MySQL, терминологията не следва йерархията „база от данни – релация – запис“, а „база от данни – колекция – документ“.

Документите в MongoDB са базирани на JSON, популярен метод за съхраняване на променливи структури от данни. JSON е акроним за JavaScript Object Notation и структурата му се състои от ключове и стойности, които могат да бъдат и вложени, което ги прави аналогични на речниците и хеш-таблиците в други езици за програмиране.

Документите се сериализират на диска във формат познат като Binary JSON (BSON) – двоично (бинарно) представяне на JSON документ. От версия 2.6 MongoDB използва стратегия за предоставяне на дисково пространство, именувана „предоставяне на размер на втора степен“ (França, 2015). Официалната документация (MongoDB, 2019d) сочи, че за разпределяне на пространството се използват степени на двойката (32 KB, 64, 128, 256, 512, … 2MB), като най-малкият възможен документ е с размер 32 байта, а размерите над 2 MB се получават като се закръгли нагоре до най-близкото кратно на числото 2. При тази организация след всеки документ има празно място, което е предпоставка за намаляване на преместванията и фрагментацията.

Колекциите и индексите се съхраняват във файлове, които първоначално са два – съответно с размери 64 и 128 MB. Целта на предварителното заемане на дисково пространство е данните в тях да заемат съседни сектори на твърдия диск, което да доведе до по-бърз достъп (Sulov, 2014). С увеличаването на данните MongoDB продължава да създава нови файлове с данни. Всеки нов файл заема двойно повече място от предхождащия го, докато размерът достигне 2 GB. В този момент всички следващи файлове са с размер 2GB. Така файлът с разширение .2 ще бъде с размер 256 MB, .3 с 512 и т.н.

След като проучихме официалната документация и друга литература, стигнахме до извода, че MongoDB може да се стигне до ситуация, в която има съществена разлика между реалното използвано пространство и заетото на твърдия диск, докато това не е възможно при MySQL.

Както и при MySQL, хранилищата са компонент, отговорен за управлението на съхраняването и достъпа до данните. Изборът на хранилище влияе върху производителността на приложението и зависи от неговите изисквания. Различни хранилища могат да бъдат избирани в рамките на различните възли от репликирана база от данни или дори различни възли на разпределена база от данни. Хранилището MMAPv1 се използва по подразбиране до версия 3.2, а за по-новите версии официалната документация (MongoDB, 2019f) препоръчва WiredTiger. Хранилището, съхраняващо цялата база от данни в оперативната памет In-Memory, е налично само в платената версия.

Хранилището на данни заема важно място в изследването, защото оказва влияние на редица характеристики на поведението на базата данни, включително и на допълнителните файловете, записващи извършените операции в базата от данни, които са източник на историческа информация. Тази информация може да бъде използвана за проследяване на тенденциите в развитието и растежа на базата от данни и по този начин да бъде симулирана нейната работа за продължителен период от време, за да се изследват характеристиките на избрана стратегия за архивиране и възстановяване, както и бързодействието на базата данни.

3. Методи за измерване размера на базата данни и тенденции в нарастването ѝ

За изчисляване на реалния размер на базата от данни в MySQL (Bradford, 2012) препоръчва да се сумират размерите на таблиците и на индексите в базата от данни. До тази информация може да бъде получен достъп чрез таблицата Tables от INFORMATION_SCHEMA (MySQL, 2019e). Според същия автор изчисляването на размера на базата от данни е от съществено значение за архивирането и възстановяването, защото размерът на получения логически архив е приблизително равен на размера на данните с разлика 10 – 15%.

Един от методите за реализиране на възстановяване към момент от време изисква да бъде поддържан списък на изпълняваните операции в СУБД. След версия MySQL 3.23.14 всички команди, които се изпълняват към базата данни, се записват в допълнителни един или повече файла. Този файл е в двоичен вид и по тази причина се нарича бинарен лог (Vaswani, 2009). За разглеждането му може да се използва инструмента mysqlbinlog, който го конвертира в текстов файл. Създаването и поддържането на лога предполага допълнително натоварване на системата, както и нужда от допълнително място на твърдия диск. Като допълнение, тъй като логовете съдържат важна информация, могат да бъдат използвани за компрометиране на сигурността и по тези причини някои администратори (Murach, 2012) предпочитат да ги изключат, което автоматично ограничава възможностите за изследване на тенденциите за развитие размера на базата данни.

По този начин стигаме до извода, че за да бъдат проследени тенденциите в развитието на базата от данни в MySQL са възможни следните подходи:

1) създаване на специфични потребителски процедури, които да записват операциите по таблиците, в случай, че наблюдение на кратък период може да осигури достоверна информация;

2) достъп до бинарния лог и анализ на наличната информация за честота на изпълнение на командите.

Архивите в MySQL могат да бъдат редактирани и с текстов редактор, което позволява да бъде изчислен брой операции изпълнявани за всяка от таблиците за определен период от време. От тази гледна точка, към горните два метода предлагаме да бъде добавен и:

3) преглед и сравнение на съществуващи архивни файлове.

Считаме, че създаването на специфични потребителски процедури може да намери изражение в създаване на тригери или инсталиране на добавки (plugins) към MySQL. Те могат да добавят собствени променливи и команди, таблици в INFORMATION_SCHEMA, да работят на фонов режим и т.н. (Schwartz, Zaitsev and Tkachenko, 2012). Пример за такава добавка е audit_log – той получава събития при изпълнението на заявки, за да може да запише какво се случва на сървъра. Audit_log е разработен от McAfee и акцентира на изисквания, свързани със сигурността и одита на бази от данни. Безплатен е и се разпространява с GNU General Public License (GitHub, 2012). Поради тази причина се използва в различни приложения – като Percona (Percona, 2019) и комерсиалната версия на MySQL (MySQL Enterprise Audit (MySQL, 2019a)), които предлагат функционалност за проследяване на възникнали събития на сървъра на база възможностите на тази добавка.

Достъп до операциите записани в бинарния лог може да бъде извършен чрез инструмента mysqlbinlog. Той дава възможност за конвертиране на двоичния файл в текстов, съдържащ SQL команди и времето за тяхното възникване (Murach, 2012). По този начин вторият и третият метод за анализ на натовареността на базата от данни добиват почти еднакво изражение от алгоритмична гледна точка – прочитане на текстов файл и извеждане на брой изпълнени операции към даден обект за единица време.

В MongoDB Влиянието на размера на файловете и реалния размер на съхраняваните в тях данни, върху скоростите на архивиране и възстановяване, може да се установи след като се избере подходящ метод за изчисление на тези размери. Директният достъп до стойностите на тези два размера е възможен чрез използване на системна команда. Резултатът от изпълнението на командата stats() е документ съдържащ различни характеристики на базата от данни в т.ч. и полетата fileSize, dataSize, IndexSize и dataSize.

Полето fileSize показва общия размер на файловете за текущата база от данни. Ако тя е току-що създадена, то това е сумата от размерите на първите два файла .0 и .1. Подкрепяме твърдението на някои автори (MLab, 2014), че разликата между dataSize и storageSize е, че първата характеристика показва реалния размер на обектите от тип BSON в базата от данни, а втората включва допълнително пространство запазено за нарастване на колекцията, както и непреразпределено изтрито пространство. IndexSize показва общия размер на индексите за базата от данни. Полето FileSize е по-голям от storageSize защото включва и екстентите използвани за индекси. За целите на изследването ще бъдат използвани и трите показателя (dataSize, StorageSize, IndexSize), като по този начин ще бъдат решени следните задачи:

- Определяне на влиянието на размера на неизползваното дисково пространство върху характеристиките на архивирането и възстановяването.
- Дефиниране на точността, с която може да се измери зависимостта между трите елемента - скорост на архивиране и възстановяване, размер на данните и размер на индексите им.

Всички операции върху данните се записват в специални журнални файлове. Ако системата спре неочаквано, записите от журналите се прилагат върху файловете с данни, когато сървърът рестартира. Те се съхраняват в директория с име journal и при запълване на последния файл се презаписва съдържанието на първия (Islam, 2011).

И двете хранилища на MongoDB имат сходен начин на работа на журналните файлове. При използване на MMAPv1 в тях се съдържат само операции, които все още не са извършени/приложени върху файловете с данни, които се прехвърлят на всеки 100 милисекунди. При WiredTiger прехвърлянето се извършва на 50, но това не гарантира изтриването им от журналния файл. Вместо това на всеки 60 секунди (или 2GB данни от журнала) се отбелязва контролна точка и се изтриват всички журнални записи, които вече са отразени във файловете с данни (MongoDB, 2019e).

От описанието на възможностите на журналните файлове става ясно, че основната им цел е да се поддържа копие на операциите от последната контролна точка до момента в случай на неочаквано спиране на сървъра, което ги прави неподходящи за целите на изследването – т.е. не могат да бъдат източник на информация за тенденциите при нарастване на базата данни.

Като алтернативен подход може да бъде използван достъп до лога на операциите (oplog). Той е специфична колекция, която съхранява хронология на логическите операции по запис в база от данни в MongoDB и според официалната документация на продукта (MongoDB, 2019d) е основен механизъм за

извършване на репликация. Този вид колекция има фиксиран размер, документите се достъпват по реда на добавяне, а при запълване на размера новите данни презаписват първите документи. Съхранява се в системната база данни Local и може да бъде достъпена като всяка друга колекция. Някои от полетата са:

- h – уникален код;
- t – точен дата и час на възникване на операцията;
- ns – комбинация от име на базата данни и колекция;
- o – код на обекта, върху който се изпълнява операцията;
- op – операция (insert, delete, create...) като единичен символ.

Имайки предвид посочените характеристики на използваните методи за запис на исторически данни от MongoDB стигаме до извода, че за проследяване на тенденциите за развитието на базата данни следва да променим определени настройки. Логът на операциите е по-подходящ източник на такава информация в сравнение с журналните файлове. Понеже е колекция с фиксиран размер (при запълване се презаписват първите документи), е непредсказуемо данни за какъв времеви интервал ще съдържа. На тази база стигаме до извода, че логът на операциите може да се използва като източник на историческа информация за тенденциите в растежа на базата от данни само ако данните от разгледания период не са по-големи от неговия размер. В противен случай размерът му следва да бъде увеличен и да се изчака събиране на данни в избрания период.

4. Методи за измерване времето за извършване на архиви

След като бяха уточнени методите за измерване размера на базата от данни и инструментите за изследване на темповете на нарастване на таблиците, е нужно да бъдат изследвани и подходите за измерване скоростта на архивиране и възстановяване.

Въпреки, че един от най-важните въпроси при стратегията за архивиране и възстановяване е да се предвиди колко време ще отнеме създаването на архива, не съществува метод, който да даде точен отговор. Причината за сложността се корени във факта, че размерът на базата от данни, обемът оперативна памет, използваното хранилище на данни, конфигурацията на MySQL, скоростта на твърдия диск и натоварването на системата имат влияние към времето за изпълнение.

Както беше посочено, за проследяване на извършените операции на сървъра може да използвана добавка за одит. Операцията по архивиране може да доведе до ограничения върху функционирането на приложенията, натоварване на системата, а самият процес може да ограничи други операции, в това число операции по поддръжка на софтуера или автоматизирано изпълнение на множество команди. За да се намали до минимум влиянието провеждане на тестове е препоръчително да не се използват допълнителни инструменти и по тази причина изключваме като възможност използването на добавки за измерване времето за изпълнение на проследяваните операции. Вместо това препоръчваме използването на командата time в Unix базирани операционни системи, като префикс на командата за архивиране в MySQL. Това ще осигури информация за точното време нужно за създаване на архива (Bradford, 2012). Ако се използва друга ОС ще трябва да се разработи приложение, изпълняващо командите по архивиране и възстановяване, в което да бъдат използвани специални маркери.

Възможностите за измерване на времето, необходимо за извършване на архивиране и възстановяване на базата от данни MongoDB ще бъдат разгледани в в две направления:

- достъп до информация, съхранявана в MongoDB, за параметрите на извършени команди;
- методи за измерване на времето за протичане на процесите по архивиране и възстановяване.

Може да се приеме, че достъпът до историческа информация съкращава процеса по обобщаването на данни и извеждането на резултати за производителността на алгоритмите за архивиране и възстановяване на разглежданата СУБД, в сравнение с извършване на измервания по време на протичането им. За постигането на този ефект обаче е нужно да съществува достатъчно надежден и удобен механизъм. Под тези характеристики предлагаме да се разбира:

1) да бъде възможен унифициран достъп до данните при различни платформи;

2) данните да бъдат надеждно съхранявани и да няма възможност да бъдат изтрити или променени – случайна или преднамерена редакция на данните за проведените тестове би обезсмислила провеждането им;

3) механизмът да предоставя точна информация – резултатите трябва да отразяват реалните характеристики на изпълняваните команди, за да бъдат достоверни и направените изводи;

4) механизмът да не оказва осезаемо влияние нито на производителността на сървъра, нито на тестовия алгоритъм.

Според официалната документация (MongoDB, 2019b) инструментът database profiler събира данни за командите и изпълняваните операции и ги записва в колекцията system.profile, която има

същите характеристики като лога на операциите. Възможностите за настройка на състоянието на инструмента са три:

- 0 – изключено – стойност по подразбиране;
- 1 – записва само операции отнели над 100 мс за изпълнение;
- 2 – записва всички операции.

По подразбиране размерът на колекцията system.profile е 1 мегабайт и при запълване новите документи презаписват най-старите. Колекция с такъв размер може да съхрани около няколко хиляди документа. В случай, че проведените тестове могат да генерират повече данни е възможно размерът на колекцията да бъде променен според нуждите, така че да не се стигне до загуба на информация.

Според цитираната официална документация database profiler има минимален ефект върху производителността на сървъра, а достъпът до колекцията се извършва със стандартните за MongoDB команди.

Като основен недостатък на този подход ще посочим необходимостта от промяна на настройки на системата за управление на базата от данни – превключване на състоянието на database profiler на включено и промяна на размера на използваната колекция. Последната операция изисква изтриване и създаване отново на system.profile, което може да доведе до конфликт с други приложения и/или задачи. По тази причина считаме, че database profiler не трябва да бъде избиран като единствено решение за тестване възможностите за архивиране и възстановяване.

Алтернативен метод за събиране на информация е характеристиките им да бъдат отчитани по време на изпълнението им. Командата cursor.explain() с параметър „executionStats“ осигурява статистики за изпълнението на заявка (MongoDB, 2019a). Тя може да предостави информация както за избрания план, така и за отхвърлените алтернативи. Тази информация включва общото време в милисекунди, необходимо за избор на план за изпълнение на заявката и нейното изпълнение (MongoDB, 2019c). При изпълнение на командите по архивиране и възстановяване с посочената функция, всяко приложение с възможност за връзка към MongoDB ще може и да има достъп до данните за времето, нужно за изпълнение на процесите на архивиране и възстановяване, и да ги съхрани по предпочитан начин в зависимост от нуждите си.

5. Резултати и заключение

Направеното изследване води до следните изводи.

1. MySQL разполага с повече възможности за избор на хранилища на данните, което увеличава гъвкавостта при разработка на приложения, но от друга страна затруднява изпълнението на административни задачи. Това налага специфичен подход в зависимост от използваното хранилище.

2. Измерването на размерите на базите от данни е по-лесно в MongoDB в сравнение с MySQL.

3. Определяне на тенденциите за развитие на базата данни въз основа на анализ на исторически данни за командите изпълнени към нея са трудна задача и в двете СУБД, като методите за провеждането й се различават съществено.

4. Измерването на времето за създаване на архив и неговото възстановяване е затруднено в MySQL, което налага да се използват решения свързани с операционната система или допълнително разработен софтуер. Този процес е значително опростен и улеснен в MongoDB

Въпреки общите черти между MySQL и MongoDB, администрирането им изправя специалистите пред различни предизвикателства. Може да се твърди, че удобството и независимостта на решенията от ОС е в полза на MongoDB, въпреки че е сравнително нова система за управление на бази от данни.

Literature

Bell, C. (2013) *Expert MySQL, Expert MySQL*. Apress. doi: 10.1007/978-1-4302-4660-2.

Bradford, R. (2012) *Effective MySQL Backup and Recovery*. McGraw Hill Professional.

DuBois, P. (2008) *MySQL*. Fourth Edi. Pearson Education.

França, W. da R. (2015) *MongoDB Data Modeling*. Packt Publishing Ltd.

GitHub (2012) *mysql-audit License*. Available at: https://github.com/mcafee/mysql-audit/wiki/License (Accessed: 10 February 2019).

Islam, R. (2011) *PHP and MongoDB Web Development Beginner's Guide*. Packt Publishing.

Kaneva, M. (2019) 'Telecommunications infrastructure and GDP/Jipp curve', *Economics and computer science*. Publishing house" Knowledge and business" Varna, (1), pp. 6–29.

MLab (2014) *How big is your MongoDB?* Available at: https://blog.mlab.com/2014/01/how-big-is-your-mongodb/ (Accessed: 10 February 2019).

MongoDB (2019a) 'Analyze Query Performance'. Available at: https://docs.mongodb.com/v3.2/tutorial/analyze-query-plan/ (Accessed: 10 February 2019).

MongoDB (2019b) 'Database Profiler'. Available at: https://docs.mongodb.com/manual/tutorial/manage-the-database-profiler/ (Accessed: 10 February 2019).

MongoDB (2019c) 'Explain Results'. Available at: https://docs.mongodb.com/manual/reference/explain-results/#explain.executionStats.executionTimeMillis (Accessed: 10 February 2019).

MongoDB (2019d) 'Glossary'. Available at: https://docs.mongodb.com/manual/reference/glossary/#term-oplog (Accessed: 10 February 2019).

MongoDB (2019e) 'Journaling'. Available at: https://docs.mongodb.com/manual/core/journaling/ (Accessed: 10 February 2019).

MongoDB (2019f) 'Storage Engines'. Available at: ttps://docs.mongodb.com/manual/core/storage-engines/ (Accessed: 10 February 2019).

Murach, J. (2012) *Murach's MySQL*. Mike Murach & Associates, Incorporated.

MySQL (2019a) 'MySQL Enterprise Audit'. Available at: https://dev.mysql.com/doc/refman/5.7/en/audit-log.html (Accessed: 10 February 2019).

MySQL (2019b) 'MySQL Glossary: ibdata file'. Available at: http://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_ibdata_file (Accessed: 10 February 2019).

MySQL (2019c) 'The ARCHIVE Storage Engine'. Available at: https://dev.mysql.com/doc/refman/5.7/en/archive-storage-engine.html (Accessed: 10 February 2019).

MySQL (2019d) 'The BLACKHOLE Storage Engine'. Available at: https://dev.mysql.com/doc/refman/5.7/en/blackhole-storage-engine.html (Accessed: 10 February 2019).

MySQL (2019e) 'The INFORMATION_SCHEMA TABLES Table'. Available at: http://dev.mysql.com/doc/refman/5.7/en/tables-table.html (Accessed: 10 February 2019).

MySQL (2019f) 'The MEMORY Storage Engine'. Available at: https://dev.mysql.com/doc/refman/5.7/en/memory-storage-engine.html (Accessed: 10 February 2019).

MySQL (2019g) 'The MyISAM Storage Engine'. Available at: https://dev.mysql.com/doc/refman/5.7/en/myisam-storage-engine.html (Accessed: 10 February 2019).

Percona (2019) 'Audit Log Plugin'. Available at: https://www.percona.com/doc/percona-server/5.6/management/audit_log_plugin.html (Accessed: 10 February 2019).

Schwartz, B., Zaitsev, P. and Tkachenko, V. (2012) *High Performance MySQL: Optimization, Backups, and Replication*. O'Reilly Media, Inc.

Sulov, V. (2014) 'On the Essence of Hardware Performance', *Research Journal of Economics, Business and ICT*, 9(1), pp. 13–18.

Vasilev, J. and KEHAYOVA-STOYCHEVA, M. (2017) 'Sales analysis by the rectangle method', *Leonardo Electronic Journal of Practices and Technologies*, (30), pp. 149–160.

Vaswani, V. (2009) *MySQL Database Usage & Administration*. McGraw Hill Professional.

Vohra, D. (2015) *Pro MongoDB™ Development, Pro MongoDB™ Development*. doi: 10.1007/978-1-4842-1598-2.

Specific administrative tasks with MongoDB and MySQL

Ivan KUYUMDZHIEV[1]

[1] University of Economics, Varna, Bulgaria
ivan_ognyanov@ue-varna.bg

Abstract. Part of software limitations depends of the database management systems in use. Database administrators are managing a number of processes on which the quality of the software depends. The publication aims to investigate and compare methods for performing important administrative tasks in the process of archiving and restoring databases. The MySQL and MongoDB database management systems have been selected for comparison. We have analyzed the similarities and differences in the categories of logical and physical data organization, methods for measuring the size of the database, its growth, as well as the time for performing backup and restore operations. Only free versions of the products are being studied as the most widely distributed products on the world market. We found some similarities in the management of both systems, as well as the strengths and weaknesses of each of them over the other. The research can be useful for database administrators.

Key words: database administration, backup, database comparison, database management systems.