УДК – 04.65

STUDY OF USING RELATIONAL AND NON-RELATIONAL DATABASES ON THE EXAMPLE OF SQL SERVER AND MONGODB

V. Bratskyi, E. Myakshylo
National University of Food Technologies

Key words:
Database
SQL
NoSQL
JSON
Relational database
Relational database
MongoDB
SQL Server

Article history:
Received 12.07.2016
Received in revised form 01.08.2016
Accepted 18.08.2016

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

ABSTRACT

A comparative analysis of the relational database with non-relational one has been conducted and their main advantages have been highlighted, which will give us an opportunity to analyze and determine when to use each type of database. Research and comparative analysis have been carried out for the MS SQL Server DBMS and MongoDB. Comparative analysis was performed on the following parameters: the rate of return query results of data entry, retrieval and correction, data storage methods, amounts of database memory, and scaling. The format of JSON documents, NoSQL, collections and documents have been considered.

ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ ЗАСТОСУВАННЯ РЕЛЯЦІЙНИХ І НЕРЕЛЯЦІЙНИХ БАЗ ДАНИХ НА ПРИКЛАДІ SQL SERVER ТА MONGODB

В.О. Брацький, О.М. М'якшило
Національний університет харчових технологій

У статті проведено порівняльний аналіз реляційної бази даних з нереляційною і виділено основні переваги кожної з них, що нададуть можливість проаналізувати й визначити, в яких випадках можна і краще використовувати нереляційні бази даних, а в яких навпаки. Дослідження та порівняльний аналіз проводились для СКБД MS SQL Server і MongoDB. Порівняльний аналіз здійснено за такими параметрами: швидкість повернення результатів запиту, внесення, вилучення та корегування даних, методи зберігання даних, обсяги пам'яті, що займає база даних, масштабованість. Розглянуто формат документів JSON, NoSQL, колекції, документи.

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

Постановка проблеми. На сьогодні наявність бази даних сама по собі не розв'язує повністю проблем організації в галузі обробки даних і прийняття рішень. Керування базою даних має здійснюватися системно, з точки зору організації в цілому, а не окремих користувачів.

Зараз, коли говорять «база даних», то мають на увазі реляційні БД. Так, мова запитів SQL використовується для роботи з реляційною моделлю даних і часто використовується для назви СУБД, наприклад, MySQL, PostgreSQL, MSSQL, Oracle та багато інших. Реляційна модель є цілісною і передбачуваною. Але це в теорії, на практиці ж часто доводиться поступатися цими принципами, надаючи перевагу продуктивності.

Крім цього, існують серйозні проблеми з масштабуванням реляційних БД. Щоб уникнути проблем з масштабуванням, використовують різні реплікації, тому найчастіше вузьким місцем веб-проектів є бази даних. Сучасні високонавантажені сайти справляються з цією проблемою за допомогою кешування, що вже стало стандартом. Цілком передбачувано з'явилися нові моделі даних, які проектувалися для того, щоб позбутися проблем реляційної моделі.

Аналіз останніх досліджень і публікацій. Порівняльний аналіз реляційної і нереляційної бази даних було проведено С. Тарасовим, який досліджував вставки даних [5]. Для тесту був обраний сценарій, що дозволяє:

- оцінити придатність СУБД для інтенсивної вставки даних від множини пристроїв;
- оцінити простоту і 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 СУБД додаток спочатку накопичив масив рядків, потім почалася транзакція, відбулася вставка з 10 рядків, який 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 випадку. 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 СУБД, або ні. У MongoDB і SQL Server ця функціональність 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.

При проведенні тесту стало зрозуміло, що обсяг використовуваної оперативної пам'яті обмежити не можна. У загальному випадку рішення полягає у створенні виділеного віртуального сервера під СУБД. Для проведення подальших тестів доводиться робити перезапуск Mongodb для очищення пам'яті. У SQL Server для аналогічного ефекту просто очищаємо буфери і кеш (DBCC). Описані дослідження формують не надто оптимістичний погляд на перспективи використання MongoDB для вирішення поставленої задачі.

З еволюцією інтернету і мобільних пристроїв значно зріс обсяг даних, які необхідно зберігати і обробляти. В наш час стає набагато складніше працювати з фіксованими структурами даних. Ще більше складнощів виникає з обробкою неструктурованих даних, особливо якщо потрібна практично необмежена масштабованість.

Метою дослідження є вибір критеріїв та розробка і застосування алгоритмів порівняння реляційних та нереляційних баз даних для формування пропозицій щодо ефективного використання кожного виду баз даних в інформаційних системах.

Виходячи з мети, були визначені такі завдання дослідження:

1. Дослідити особливості побудови реляційних і нереляційних баз даних.

2. Дослідити основні аспекти використання реляційних і нереляційних баз даних, визначити критерії їх порівняння та методи оцінювання ефективності використання досліджуваних баз даних.

3. Ознайомитися з особливостями побудови, використання та характеристиками нереляційної СУБД MongoDB.

4. Розробити методику тестування реляційної та нереляційної бази даних для порівняння їх можливостей.

5. Провести тестові випробування та порівняти реляційну і нереляційну базу даних за визначеними критеріями

6. На основі проведених досліджень сформулювати пропозиції щодо використання реляційних і нереляційних баз даних в інформаційних системах.

Викладення основних результатів дослідження. *Архітектура реляційних БД.* У наш час як стандартна архітектура БД використовується трирівнева система організації, запропонована американським комітетом із стандартизації ANSI [5]. Система складається із зовнішньої моделі (схеми) даних, внутрішньої схеми даних (концептуальної моделі) та безпосередньо фізичної БД (рис. 1).

Перший рівень визначає точку зору на БД різних додатків користувачів. Кожен додаток бачить і обробляє тільки ті дані, які потрібні певному користувачеві. Так, концептуальний рівень — центральна управляюча ланка, яка представляє у найбільш загальному вигляді, що об'єднує дані, використовувані всіма додатками, що працюють з базою даних. Фізичний рівень — дані, розташовані у файлах або сторінкових структурах, що знаходяться на зовнішніх носіях інформації.

Така архітектура дозволяє забезпечити логічну (між рівнями 1 і 2) та фізичну (між рівнями 2 і 3) незалежність при роботі з даними. Логічна незалежність передбачає можливість зміни одного додатка без коригування іншого. Фізична незалежність передбачає можливість перенесення інформації, що зберігається, з одних носіїв на інші за умови збереження працездатності всіх додатків, що працюють з цією базою даних.

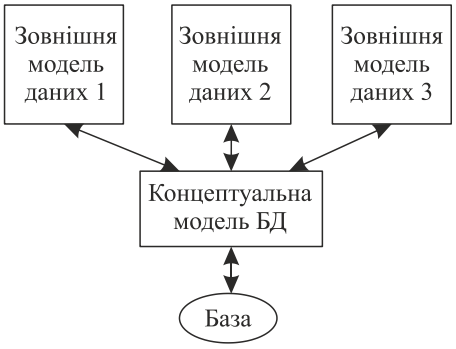

Рис. 1. Архітектура бази даних за стандартом ANSI

Розробка структури БД — найважливіше завдання, яке вирішується при проектуванні БД. Структура БД (таблиці і зв'язки між ними) — це одне з основних проектних рішень при створенні додатків з використанням БД. Створена розробником структура БД описується на мові визначення даних СУБД.

Будь-яка СУБД дозволяє виконувати такі операції з даними:

- додавання записів у таблиці;
- видалення записів з таблиці;
- оновлення значень деяких полів в одній або декількох записах у таблицях БД;
- пошук одного або декількох записів, що задовольняють задану умову.

Для виконання цих операцій застосовується механізм запитів. Результатом виконання запитів є відібрана за певними критеріями безліч записів або зміни в таблицях. Запити до бази формуються на спеціально створеній для цього мові, що називається «мова структурованих запитів» (SQL — Structured Query Language).

Нереляційні бази даних (NoSQL). Протягом останнього десятиліття розробники і системні адміністратори завжди надають перевагу реляційним СУБД. Незважаючи на те, що вони не такі вже й адаптивні, функціонал реляційних СУБД дозволяє створювати досить складні системи даних. Цього було більш ніж достатньо, поки не з'явилися NoSQL СУБД.

NoSQL, або нереляційні бази даних об'єднують нереляційні сховища даних, які не підкоряються звичним правилам зберігання даних, так званому ACID (atomicity, consistency, isolation, durability). Зазвичай такі системи не мають жорсткої структури і не використовують таблиць для збереження даних і зв'язків між ними.

NoSQL розшифровується як Not Only SQL — не тільки SQL. Це рух не проти SQL, а за те, щоб використовувати його тільки там, де це потрібно. Нереляційні бази даних поділяють на кілька типів залежно від їх масштабованості, моделі даних і запитів, а також систем зберігання даних.

Останнім часом термін «NoSQL» став дуже модним і популярним, активно розвиваються і просуваються всілякі програмні рішення під цією вивіскою. Синонімом NoSQL стали величезні обсяги даних, лінійна масштабованість, кластери, кросплатформнність.

Термін «NoSQL» з'явився близько десяти років тому. Спочатку це була назва чергової реляційної системи управління базами даних. Проте передбачала вона трохи інше — уникнути використання стандартів SQL. У наступні кілька років інші підхопили ідею і почали використовувати цей термін стосовно нереляційних баз даних. За задумом NoSQL бази даних і СУБД не передбачають внутрішніх зв'язків. Вони не ґрунтуються на одній моделі, а кожна база даних залежно від цілей використовує різні моделі.

Існує досить багато різних моделей і функціональних систем для NoSQL баз даних:

- сховище ключ-значення — Redis, MemcacheDB тощо (зазвичай зберігають дані в пам'яті);
- розподілене сховище (Column-oriented) — Cassandra, HBase тощо (призначені для дуже великих обсягів даних);
- документо-орієнтовані СУБД — MongoDB, Couchbase тощо (призначені для зберігання ієрархічних структур даних — документів);
- БД на основі графів — OrientDB, Neo4J тощо.

MongoDB [4] реалізує новий підхід до побудови баз даних, де немає таблиць, схем, запитів SQL, зовнішніх ключів і багатьох інших речей, які притаманні об'єктно-реляційним базам даним.

Зазвичай всі дані зберігалися в реляційних базах даних (MS SQL, MySQL, Oracle, PostgresSQL). При цьому не зважали, чи підходять реляційні бази даних для зберігання даного типу даних, чи ні. На відміну від реляційних баз даних, MongoDB пропонує документо-орієнтовану модель даних, завдяки чому MongoDB працює швидше, має кращу масштабованість, її легше використовувати. Але навіть враховуючи всі недоліки традиційних баз даних MongoDB, важливо розуміти, що завдання, як і методи їх вирішення, бувають різні. В якійсь ситуації MongoDB дійсно поліпшить продуктивність програми, наприклад, якщо треба зберігати складні за структурою дані. В іншій же ситуації краще буде використовувати традиційні реляційні бази даних. Крім того, можна використовувати змішані підхід: зберігати один тип даних у MongoDB, а інший тип даних — у традиційних БД.

Вся система MongoDB може представляти не тільки одну базу даних, що знаходиться на одному фізичному сервері. Функціональність MongoDB дозволяє розташувати кілька баз даних на декількох фізичних серверах, і ці бази даних зможуть легко обмінюватися даними й зберігати цілісність.

Формат даних у MongoDB. Одним із популярних стандартів обміну даними та їх зберігання є JSON (JavaScript Object Notation). JSON ефективно

описує складні за структурою дані. Спосіб зберігання даних у MongoDB в цьому плані схожий на JSON, хоча формально JSON не використовується. Для зберігання в MongoDB застосовується формат, який називається BSON (Бісон) або скорочення від binary JSON.

BSON дозволяє працювати з даними швидше: швидше виконується пошук і обробка. Хоча треба зазначити, що BSON, на відміну від зберігання даних у форматі JSON, має невеликий недолік: в цілому дані в JSON-форматі займають менше місця, ніж у форматі BSON, з іншого боку, даний недолік нівелюються швидкістю.

Кросплатформність. MongoDB написана на C ++, тому її легко перенести на будь-які платформи. MongoDB може бути розгорнута на платформах Windows, Linux, MacOS, Solaris. Можна також завантажити вихідний код і скомпілювати MongoDB, але рекомендується використовувати бібліотеки з офсайта.

Якщо реляційні бази даних зберігають рядки, то MongoDB зберігає документи. На відміну від рядків, документи можуть зберігати складну за структурою інформацію. Документ можна представити як сховище ключів і значень. Ключ являє просту мітку, з яким асоційований певний масив даних.

Однак, незважаючи на всі відмінності, є одна особливість, яка зближує MongoDB і реляційні бази даних. У реляційних СУБД зустрічається таке поняття як первинний ключ, що описує певний стовпець, який має унікальні значення. У MongoDB для кожного документа є унікальний ідентифікатор, який називається _id. І якщо явно не вказати його значення, то MongoDB автоматично згенерує для нього значення.

Кожному ключу надається певне значення, хоча треба враховувати одну особливість: якщо в реляційних базах чітко окреслена структура, де є поля, і якщо якесь поле не має значення, йому (залежно від налаштувань конкретної БД) можна присвоїти значення NULL. У MongoDB все інакше. Якщо якомусь ключу не надано значення, то цей ключ просто опускається в документі і не вживається.

У традиційному SQL є таблиці, в MongoDB — колекції. І якщо в реляційних БД таблиці зберігають однотипні жорстко структуровані об’єкти, то колекції можуть містити найрізноманітніші об’єкти, що мають різну структуру і різний набір властивостей:

- база даних — набір колекцій;
- колекція — набір документів;
- документ — набір полів;
- поле — пара ключ:значення.

Реплікація — це процес синхронізації даних між декількома серверами. Реплікація забезпечує надмірність і збільшує доступність даних. З декількох копій даних на різних серверах баз даних реплікація захищає бази даних від втрати на одному сервері. Реплікація також дозволяє відновлювати після збою апаратного забезпечення й обслуговування переривань. За допомогою додаткових копій даних можна виділити один для аварійного відновлення, звітності, або резервного копіювання.

Система зберігання даних у MongoDB являє собою набір реплік. У цьому наборі є основний вузол, а також може бути набір вторинних вузлів. Усі вторинні вузли зберігають цілісність і автоматично оновлюються разом з

оновленням головного вузла. І якщо основний вузол з якихось причин виходить з ладу, то один із вторинних вузлів стає головним.

Шардінг — це спосіб для зберігання даних на декількох машинах. MongoDB використовує шардінг для підтримки розгортання з дуже великими наборами даних і високою пропускною операцій.

MongoDB реалізувала горизонтальне масштабування за принципом «автоматичного шардингу». Шардінг поширює дані на фізичні машини, які можуть обслуговувати дані паралельно. Розподіл навантаження на шарди надає можливість базі даних обслуговувати більше запитів, розподіляючи навантаження пропорційно на кожну машину. Із збільшенням кластера кожний шард обробляє менше операцій. У результаті кластер може збільшити продуктивність шляхом горизонтального росту.

Шардінг, або горизонтальне масштабування, навпаки, ділить набір даних і розподіляє дані по декількох серверах або створює «осколки». Кожен осколок є як самостійною базою даних, так і колективною, оскільки всі дані становлять єдину логічну базу даних. Шардінг взаємодіє з проблемою масштабування, щоб підтримувати високу пропускну здатність і великі набори даних:

- шардінг зменшує кількість операцій. Кожен осколок обробляє менше операцій. У результаті кластер може збільшити пропускну здатність і по горизонталі. Наприклад, щоб вставити дані, додаток повинен отримати доступ до осколка, відповідального тільки за цей запис;

- шардінг зменшує кількість даних, які зберігає сервер. Кожен осколок має менше даних, тоді як кластер зростає. Наприклад, якщо база даних має набір даних 1 терабайт і є 4 осколки, то кожен уламок може провести тільки 256 Гб даних. Якщо є 40 осколків, то кожен уламок може провести тільки 25 ГБ даних.

MongoDB забезпечує збалансований кластер, використовуючи два фонових процеси: розщеплення і балансування.

Розщеплення — це фоновий процес, який стримує фрагменти від надмірного зростання обсягу даних. Коли фрагмент перевищує межі, MongoDB розбиває його навпіл. Відбувається вставка і поновлення тригерів розщеплення даних.

Балансування — фоновий процес, який управляє міграцією шарда (обсяг даних на одному з комп'ютерів кластера). Балансир може працювати в будь-якому з маршрутизаторів запитів кластера. Коли дані в одному шарді перевищують заданий розмір, то Mongodb розділяє колекцію на фрагменти. Кожен фрагмент, за стандартом, має розмір 64 мб. Коли колекція перевищує 64 мб, Mongodb починає ділити її на дрібніші фрагменти. При цьому навантаження на сервер не відбувається, тому що балансування здійснюється за один раз.

Відсутність жорсткої схеми бази даних і в зв'язку з цим потреби при щонайменшій зміні концепції зберігання даних перебудовувати цю схему значно полегшують роботу з базами даних MongoDB і подальшим їх масштабуванням. Крім того, розробникам, не потрібно витрачати час на побудову складних запитів.

Однією з проблем при роботі з будь-якими системами баз даних є збереження даних великого розміру. Можна зберігати дані у файлах, використовуючи різні мови програмування. Деякі СУБД пропонують спеціальні типи даних для зберігання бінарних даних у БД (наприклад, BLOB у SQL Server).

На відміну від реляційних СУБД, MongoDB дозволяє зберігати різні документи з різним набором даних, однак при цьому розмір документа обмежується 16 мб. Але MongoDB пропонує рішення — спеціальну технологію GridFS, яка дозволяє зберігати дані за розміром більше, ніж 16 мб.

Система GridFS складається з двох колекцій. У першій колекції, яка називається files, зберігаються імена файлів, а також їх метадані, наприклад, розмір, в іншій колекції, яка називається chunks, у вигляді невеликих сегментів зберігаються дані файлів, зазвичай сегментами по 256 кб. Для тестування GridFS можна використовувати спеціальну утиліту mongofiles, яка є в пакеті mongodb.

Порівняння продуктивності MongoDB та SQL SERVER.

Для визначення інших порівняльних характеристик MongoDB і СУБД SQL Server було проведено власні дослідження. Так, швидкість роботи з обома видами БД перевірялася за таким алгоритмом:

1. Спочатку проводилася масова вставка 1 млн записів в одному потоці і визначався час проведення операції.

2. Потім перевірялася робота з базою даних із 10 потоків. В кожному потоці виконувалася така робота:

2 % — додавання нових записів;

3 % — оновлення випадкового запису;

1 % — видалення випадкового запису;

13 % — вибірка випадкової сторінки з 20 записів;

81 % — вибірка однієї випадкової записи.

Для порівняння швидкодії використовувалася SQL Server 2012 Enterprise. Результати першої операції: вставка 1 млн записів на SQL Server тривала 3040 с (~ 50 хв), така ж процедура на MongoDB зайняла 214 с (~ 3,5 хв), що в 14 разів швидше. До речі, розмір бази даних: SQL Server — 96 Мб, MongoDB — 384 Мб. Різниця в 4 рази, причому не на користь MongoDB. Чим менший файл, тим менше операцій введення-виведення, тим простіше його кешувати, тим швидше робити резервні копії.

Звичайно, наведений тест некоректний. Вставляти 1 млн записів у SQL-сервер по одному — це абсолютно невірне рішення, необхідно використовувати оптимізацію. Масова вставка 1 млн записів за допомогою оператора Bulk Insert теж займає близько 3 хв, однак не завжди в програмі можна застосувати оператор масової вставки. Також можна вставляти дані «порціями» по 1000 записів в одній транзакції — це теж істотно прискорює вставку.

Проведемо другий тест, який набагато більше підходить до реальної роботи з типовим додатком: 3 вставки за різною кількістю елементів, 3 — видалення по індексам, конкретним значенням, 5 правок — по всьому стовпцю, індексом, заданими параметрами, з перевіркою заданої умови і 15 різних вибірок — посторінкових і до одного запису, причому робота ведеться одночасно з декількох потоках.

Таблиця 1. **Результат вставки 1 млн записів у БД**

	MongoDB	SQL Server
1 потік	1 с	17 с
10 потоків	3 с	211 с

На простих операціях CRUD MongoDB демонструє дуже серйозний приріст продуктивності (табл. 1). Однак більш глибоке вивчення аналітичних функцій MAX / MIN, AVG, SUM, COUNT, DISTINCT, GROUP BY показує високу трудомісткість в їх реалізації, особливо на великих базах.

Висновок

Завдяки значному прогресу в розвитку об'єктної технології за останні п'яти років виробникам вдалося довести ООСУБД до такого рівня, що вони стали цілком відповідати реальним вимогам ринку. Технологія об'єктних СУБД дозріла для великих проектів, проте для дійсно масового її поширення необхідний спеціальний інструментарій. Відчувається потреба в інтеграції ООСУБД з існуючими інструментальними засобами. Розроблювачі вже сьогодні можуть продуктивно використовувати C#, Java, C++ чи Python, що підтримують ООСУБД, хоча більшість продуктів для створення додатків є об'єктно-орієнтовними, але працюють, як і раніше, з реляційними БД.

Отже, проведене дослідження доводить, що нереляційні БД MONGODB:

- зручні при зберіганні неструктурованої інформації;
- масштабуються набагато краще за RDBMS і більш ефективні при обробці дуже великих обсягів даних;
- ефективні для аналітичної обробки даних;
- часто не мають транзакційних механізмів.

Література

1. Реляционные базы данных обречены? [Електронный ресурс]. — Режим доступа: http://habrahabr.ru/post/103021/.

2. *Banker K.* MongoDB in Action / K. Banker. — London, 2011. — 312 p.

3. *Фаулер М.* NoSQL: новая методология разработки нереляционных баз даннях / М. Фаулер, П. Кумар, Дж. Садаладж. — Москва: Вильямс, 2014. — 192 с.

4. Возможности MongoDB [Електронный ресурс]. — Режим доступа: http://habrahabr.ru/post/119703/.

5. *М'якшило О.М.* Організація баз даних та знань: навчальний посібник / О.М. М'якшило. — Київ, НУХТ, 2013. — 148 с.

6. Офіційний сайт MongoDB [Електронний ресурс]. — Режим доступу: https://www.-mongodb.org/.

7. *Айтхожаева Е.Ж.* Стандартный язык баз данных SQL. Учебное пособие. — Алматы, 2005. — 48 с.

8. *Dzhurenko T.* Analysis of Text Mining methods in Web search / T. Dzhurenko, O. Myakshylo, G. Cherednichenko // Ukrainian Food Journal. — 2015. — V. 4. — I. 3. — P. 508—519.

ИССЛЕДОВАНИЕ ОСОБЕННОСТЕЙ ИСПОЛЬЗОВАНИЯ РЕЛЯЦИОННЫХ И НЕРЕЛЯЦИОННЫХ БАЗ ДАННЫХ НА ПРИМЕРЕ SQL SERVER И MONGODB

В.О. Брацкий, Е.М. Мякшило
Национальный университет пищевых технологий

В статье проведен сравнительный анализ реляционной базы данных с нереляционной и выделены основные преимущества каждой из них, которые

дадут возможность проанализировать и определить, в каких случаях лучше использовать реляционные и нереляционные базы данных. Исследования и сравнительный анализ проводились для СУБД MS SQL Server и MongoDB. Сравнительный анализ был осуществлен по следующим параметрам: скорость возврата результатов запроса, внесение, удаление и корректировка данных, методы хранения данных, объемы памяти, которые занимает база данных, масшабирование. Рассматривался формат документов JSON, NoSQL, коллекции, документы.

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