Сога Д.С.

студентка фізико-математичного факультету
Науковий керівник - А.Л. Федорчук
кандидат педагогічних наук,
доцент кафедри прикладної математики та інформатики
Житомирський державний університет імені Івана Франка

NOSQL – НОВА МЕТОДОЛОГІЯ РОЗРОБКИ НЕРЕЛЯЦІЙНИХ БАЗ ДАНИХ

Величезне зростання нових додатків, пов’язаних зі зберіганням і обробкою великої кількості даних додали більше проблем у використанні

реляційних баз даних. Класичні системи SQL перестали задовольняти всі необхідні потреби. Це призвело до створення нової моделі бази даних під назвою NoSQL. Тому основною метою статті є визначення особливостей NoSQL баз даних, принципів їх роботи та коротке порівняння з моделлю MySQL.

Основою бази даних у NoSQL є хеш-функція – математичний алгоритм, який перетворює вхідні дані будь-якого розміру у вихідні дані фіксованої довжини. Ключ кожної пари "ключ/значення" створюється за допомогою хешування, і цей хеш-код використовується для скерування пари до NoSQL сервера, де вона буде зберігатись для пошуку в майбутньому [1].

Для пошуку пари "ключ/значення" потрібно надати базі даних сам ключ.

Після прочитання опису цього процесу може постати питання: "Як користувач чи програма виконує складніші запити, такі як пошук ключів, які мають якесь конкретне значення, або сортування по значенню?" Відповідь проста – NoSQL просто не підтримує функціоналу такого типу. NoSQL створений для ефективного зберігання пар "ключ/значення" зі швидким доступом до них, коли програмі просто потрібно місце для схову даних з подальшим пошуком їх тільки за ключем.

NoSQL відноситься до бази даних, яка не базується на мові SQL (Structured Query Language), що часто асоціюється з базами даних. Насправді, NoSQL дані не є реляційними, NoSQL БД зазвичай не мають схем й у них більш узгоджена модель, ніж в традиційних реляційних базах даних.

Термін "NoSQL" означає, що традиційні реляційні БД не можуть вирішити всі завдання, особливо ті, які пов'язані з великими обсягами даних. Термін був розширений до значення "Not only SQL", який означає підтримку для потенційних SQL-інтерфейсів в кожному ядрі нереляційних БД. Розробники додатків, які використовують NoSQL рішення, не обов'язково виключають реляційні БД, а замість цього бачать цінність у правильності використання кожного зі сховищ даних для вирішення відповідного завдання [2].

Кешування результатів в NoSQL є спільним завданням підвищення чуйності додатка. Наприклад, web-сайт віддає одні й ті ж відповіді сотням і тисячам користувачів. Замість того, щоб перераховувати в реляційній БД одне і те ж, варто вручну налаштувати кешування. Деякі NoSQL сховища надають схожі рішення, але розробнику не потрібно підтримувати користувальницький кеш.

Деякі NoSQL БД зберігають пари "ключ/значення" для швидкого пошуку, наприклад, у разі доступу "запит/відповідь" [3].

Майже всі "key-value" бази даних мають вигляд розподілених хеш-таблиць. Це сама сильна їх сторона, адже це забезпечує високу

продуктивність і сильно спрощує масштабування. Тим не менш, ця властивість спливає на поверхню, коли необхідно працювати зі списками.

З MySQL наступні завдання вирішуються дуже просто:

- вибрати 10 останніх користувачів;
- вибрати найпопулярніші пости в блозі;
- знайти продукти, для яких залишено більше 3-х коментарів;
- повнотекстовий пошук;
- пошук по будь-якому властивості (знайти користувача по e-mail).

Але в базі ключ-значення подібні задачі часто викликають труднощі. Ці бази даних зовсім не призначені для цих задач. Але це тільки на перший погляд, адже формулювати завдання можна по різному. Стратегія розв'язку буде залежати від конкретної ситуації.

Існує клас задач, коли потрібно зробити вибірку одного об'єкту не за первинним, а вторинному ключу (наприклад, пошук користувача по e-mail, пошук автора за нікнеймом і т.д.). Принцип вирішення цієї задачі наступний: після створення нового об'єкту, потрібно створити посилання на його первинний ключ для всіх його властивостей, за якими потрібно буде робити вибірку. Наприклад створюємо користувача:

user_111: {
    name: Ronald,
    mail: bmth9@ukr.net
}

Дублюємо ключ email:

user_email_:bmth9@ukr.net: {
    id: 111}

Тепер, ми можемо робити вибірку даних користувача за його поштовою адресою в два етапи.

Інші типи даних більш документоорієнтовані та мають різні варіації. Наприклад, форми даних можуть мати додаткові поля. Реляційні БД з їх строгими схемами висувають вимоги до всіх полів кожного збереженого рядку даних. Документоорієнтовані NoSQL сховища більш гнучкі й ефективні в цьому плані [5].

Щодо швидкого доступу до великих наборів даних, то реляційні БД втрачають продуктивність при пошуку в великих обсягах даних. Історично, розробники будують системи, в яких пишуться SQL запити для знаходження невеликої кількості записів, які видаляють для збільшення загальної ефективності. Чим більш результуючий набір, тим дорожчим стає запит. Великі обсяги даних або запити, які включають обробку великих обсягів даних, називаються "data warehousing" (сховища даних) [4].

NoSQL також вважається альтернативою традиційним реляційним БД, тому що деякі з вимог до узгодженості, які є частиною реляційних БД, дуже відрізняються в сучасних системах.

Розробники знають, що деякі вимоги до даних не вимагають жорсткої ACID моделі реляційних БД, але вони, як правило, мають гіршу продуктивність. Натомість вони можуть задовольнити свої потреби, використовуючи узгодженість в кінцевому рахунку, яка, як правило, має кращу продуктивність. Деякі NoSQL сховища навіть надають розробникові вибір якою має бути узгодженість, жорсткою чи ні.

Використовуючи ту чи іншу базу даних, необхідно враховувати обмеження, які можуть виникати при роботі з нею.

SQL є потужним, 40-річним стандартом, який був можливий тому, що всі реляційні БД мали одну і ту ж концепцію збереження даних в таблиці та посилання на них за допомогою зовнішнього ключа. Незважаючи на те, що перехід з однієї реляційної БД на іншу не на 100% прозорий, він набагато легше, ніж перехід між двома різними NoSQL сховищами.

Тому кожне NoSQL сховище має унікальний підхід до того, як зберігати дані, а також як різні біти даних відносяться до інших, немає одного API для управління всім цим. Коли застосовується нове NoSQL сховище, розробник повинен витратити час і зусилля на вивчення нової мови запитів, а також семантику узгодженості.

Доступно безліч NoSQL сховищ та нижче представлені найбільш популярні:

- **MongoDB** – документна БД з відкритим вихідним кодом.
- **CouchDB** – БД, яка використовує JSON для документів, JavaScript для MapReduce запитів, і звичайний HTTP для API.
- **GemFire** – розподілена платформа керування даними, що забезпечує динамічну масштабованість, високу продуктивність і збереження як у БД.
- **Redis –** сервер структур даних, де ключами можуть бути рядки, хеші, списки, набори та сортовані набори.
- **Cassandra** – БД, яка забезпечує масштабованість і високу надійність без втрати продуктивності.
- **Memcached –** високопродуктивна, розподілена в пам'яті й об'єктна система кешування з відкритим вихідним кодом.
- **Hazelcast** – високомасштабуюча розподілена платформа з відкритим вихідним кодом.
- **Neo4j –** високопродуктивна, enterprise-класу графова БД з відкритим вихідним кодом.

NoSQL сховища стають більш популярними та протестовані в багатьох ситуаціях. Ці ситуації включали великі обсяги даних, а також великий темп зростання даних у багатьох системах, але враховуючи обмеження NoSQL, вони ніколи повністю не замінять реляційні бази даних. Проте, вони мають перспективне майбутнє щодо простого зберігання даних високоефективним способом.

Список використаних джерел:

1. NoSQL базы данных. Понимаем суть [Електронний ресурс] Режим доступу: https://habrahabr.ru/post/152477/

2. Understanding SQL And NoSQL Databases And Different Database Models.[Електронний ресурс] Режим доступу: https://www.digitalocean.com/community/tutorials/understanding-sql-and-nosql-databases-and-different-database-models

3. Понимание NoSQL [Електронний ресурс] Режим доступу: http://spring-projects.ru/understanding/nosql/

4. Реляционная база данных. [Електронний ресурс] Режим доступу: https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85

5. Решения на NoSQL базах [Електронний ресурс] Режим доступу: http://ruhighload.com/index.php/2010/06/03/nosql-%D0%BF%D0%BE%D0%B4%D1%85%D0%BE%D0%B4%D1%8B-%D0%BA-%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D1%8E-%D1%82%D0%B8%D0%BF%D0%B8%D1%87%D0%BD%D1%8B%D1%85-%D0%B7%D0%B0%D0%B4%D0%B0%D1%87/