системи, способу їх представлення та формування. Залишається актуальним питання створення та розробки нових нейромережевих структур та нечітких регуляторів, що будуть орієнтовані на підтримку інтелектуальних технологій обробки інформації та керування. І останнє, подальший розвиток інтелектуалізованих технологій на різних рівнях системи, чи то на виконавчому рівні (інтелектуальний привід), чи то на рівні організації поведінки системи, дозволить забезпечити створення нових механізмів та пристроїв, що матимуть високі технічні характеристики та функціональні можливості.

Нечітка логіка є перспективним напрямком сучасної теорії керування. Вона забезпечує принципово нових підхід до проектування систем керування, можливість вирішення широкого кола проблем, в яких дані, цілі та обмеження являються дуже складними або невизначеними та не піддаються класичній теорії керування.

Список використаних джерел:

1. Рутковская Д., Пилинский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы – М.: Горячая линия-Телеком, 2006 – 452 с.

2. Васильев В. И., Ильясов Б. Г. Интеллектуальные системы управления. Теория и практика: учебное пособие – М.: Радиотехника, 2009 – 392 с.

3. Интеллектуальные системы автоматического управления / Под ред. И. М. Макарова, В. М. Лохина – М.: ФИЗМАТЛИТ, 2001 – 576 с.

Койда В.І.

студент,
Національний технічний університет України
«Київський політехнічний інститут»

ОСОБЛИВОСТІ НЕРЕЛЯЦІЙНИХ БАЗ ДАНИХ NOSQL, ЇХ ПЕРЕВАГИ ТА НЕДОЛІКИ У ПОРІВНЯННІ З SQL

NoSQL охоплює широкий спектр різних технологій баз даних (БД), які були створені у відповідь на вимоги, що висуваються при розробці сучасних програмно-технічних комплексів та систем. Розробники програмного забезпечення працюють з додатками, які генерують величезні обсяги нових, швидко змінюваних даних – структурованих, слабоструктурованих, неструктурованих і поліморфних даних. Програми, що раніше встановлювалися у кінцевих користувачів, тепер перетворилися на сервіси, які повинні бути постійно доступні з багатьох різних пристроїв та глобально масштабуватися для обслуговування мільйонів користувачів. Організації та підприємства тепер використовують масштабовані архітектури на основі відкритого програмного забезпечення, кластерні сервери й хмарні обчислення замість великих монолітних серверів та жорсткої інфраструктури зберігання.

На сьогодні існують такі основні типи баз даних NoSQL [1]:

– документо-орієнтовані, зберігають ключ у парі з складною структурою даних, що відома як документ. Документи можуть містити багато різних пар ключ-значення, пар ключ-масив, або навіть вкладені документи.

– графо-орієнтовані, використовуються для зберігання інформації про мережі даних у вигляді графів. Яскравим прикладом таких структур є соціальні мережі.

– ключ-значення, найпростіший тип баз даних NoSQL. Кожен елемент в базі даних зберігається в якості імені атрибута або ключа, разом з його значенням.

– стовпцеві, такі як Cassandra і HBase, оптимізовані для запитів над величезними наборами даних, які зберігаються у вигляді стовпців, а не рядків.

Однією із ключових переваг нереляційних БД є наявність динамічної схеми БД [2]. Реляційні БД вимагають, щоб схеми були чітко визначені до того, як дані будуть додані. Наприклад, якщо ви хочете зберігати інформацію про клієнтів, таку як прізвище, ім'я, номер телефону, електронна пошта, адреса, місто, країна, то необхідно заздалегідь вказати відповідні атрибути в SQL БД.

Такий підхід погано підходить до популярної моделі розробки програмного забезпечення agile, тому що кожен раз, коли додається новий модуль або функціонал, необхідно змінювати схему БД. Наприклад, якщо ви вирішите через кілька ітерації розробки зберігати улюблені товари клієнтів, у додачу до їх телефонів та іншої інформації, вам необхідно буде додати ще один атрибут до схеми БД, а потім перенести всю базу на нову схему. Якщо БД велика, то це дуже повільний процес, який вимагає значних ресурсів. Якщо частота таких ітерації досить велика, то сумарний час процесу міграції може бути великим. Також, для реляційних БД не існує способів ефективної обробки неструктурованих даних, чи даних, структура яких заздалегідь невідома.

NoSQL системи побудовані таким чином, щоб дозволити введення даних без заздалегідь визначеної схеми. Це значно спрощує внесення значних змін до програми та даних в режимі реального часу, не турбуючись про переривання обслуговування – це означає, що розробка програми відбувається швидше, код є більш надійним, і потрібно менше часу для адміністрування БД. Розробникам, як правило, доводилося прописувати у коді перевірки, для забезпечення контролю якості даних, такі як наявність специфічних атрибутів, типи даних і допустимі значення. Використовуючи більш глибокі налаштування NoSQL, можна застосувати такі перевірки в самій БД, що звільняє розробника від зайвої роботи та залишає переваги динамічної agile розробки.

Не менш важливою властивістю є автосегментація та масштабованість [2]. Структура реляційних БД принципово передбачає розширення та масштабованість по вертикалі, тобто використовується один сервер для збереження усіх даних. Цей сервер повинен бути достатньо потужним для виконання операцій декартового добутку та обробки транзакцій. Зі збільшенням об'ємів інформації необхідно удосконалювати цей сервер, що швидко стає дорогим для обслуговування та експлуатації, але навіть після удосконалення, обмеження на об'єми даних залишаються. Гнучким рішенням для підтримки швидкозростаючих додатків це масштабування в ширину, тобто

додавання нових серверів для збереження даних, ніж зосередження більшої ємності та потужності на одному сервері.

Сегментування бази даних серед багатьох серверів у деяких випадках може бути досягнуто і з базами даних SQL, використовуючи мережі збереження даних (SAN) та інші складні структури, що поєднують апаратне забезпечення у єдиний логічний сервер.. Оскільки в реляційній БД така можливість не вбудована спочатку, то командам розробників необхідно розгортати декілька екземплярів БД на певній кількості машин. В такому випадку дані зберігаються в кожному екземплярі автономно. Як наслідок, розробникам необхідно розробити складну архітектуру додатку для роботи з розподіленими даними, розподіленими запитами та агрегацією результатів з усіх екземплярів БД. Також, необхідно розробити механізми для обробки помилок ресурсів, для виконання операції об'єднання між різними екземплярами БД, для балансування навантаження та відновлення даних. Крім цього, при такому способі сегментування майже неможливо забезпечити цілісність транзакцій реляційної БД.

З іншого боку, NoSQL, як правило, має вбудовану підтримку автосегментації, для автоматичного розгортання БД на довільній кількості серверів. При цьому додаткам не потрібно знати склад пулу серверів. Дані та запити автоматично балансуються між серверами, і коли один з серверів виходить з ладу, його можна непомітно замінити іншим, не порушуючи роботу додатку.

Більшість NoSQL баз даних також підтримують автоматичне відновлення БД для забезпечення доступності, у разі виникнення збоїв або проведення планових робіт технічного обслуговування. Більш складні NoSQL БД мають функцію абсолютного самовідновлення, пропонуючи тим самим автоматизоване аварійне перемикання і відновлення, а також можливість розміщення бази даних у кількох географічних регіонах. На відміну від реляційних БД, NoSQL, як правило, не потребують окремих дорогих доповнень, компонентів чи додаткових модулів для здійснення відновлення даних.

Незважаючи на те, що бази даних NoSQL мають ряд істотних переваг, вони також мають цілий ряд недоліків [3]. Хоча ці проблеми стосуються більше підприємств, аніж розробників, та все ж вони заслуговують уваги.

Багато NoSQL систем ще знаходяться в розробці на стадіях бета версій та не мають деяких ключових функцій. Тому слід проявляти обережність при прийнятті рішення про те, чи варто використовувати цей тип БД.

Підприємства, як правило, покладаюся на запевнення в тому, що якщо ключова системи вийде з ладу, то вони будуть мати можливість отримати своєчасну і кваліфіковану підтримку. Більшість NoSQL систем – це проекти з відкритим вихідним кодом. Ці системи створювалися невеликими компаніями або стартапами, які не мають глобального охоплення та великих ресурсів для забезпечення підтримки користувачів на високому рівні, на відміну від постачальників поширених реляційних БД таких як Oracle.

Бази даних NoSQL пропонують невеликий функціонал для створення запитів та аналізу даних, оскільки вони не використовують SQL. Запити, що є простими для реляційних БД, вимагають значних навичок програмування при

використанні нереляційних БД. Крім того, широко використовувані інструменти бізнес аналітики на даний момент не мають сумісності з NoSQL.

Хоча завданням розробки NoSQL було створення БД, що вимагає мінімального адміністрування, на даний момент цього ще не досягнуто. Ці бази даних вимагають значного рівня навичок і зусиль для встановлення, розгортання та адміністрування БД.

Більшість розробників NoSQL змушені постійно навчатися, оскільки ця технологія є відносно молодою. Згодом, ситуація зміниться і професіоналів в області нереляційних БД стане більше, але на даний момент легше знайти спеціаліста з реляційних БД, аніж експерта в NoSQL.

NoSQL технологія набирає популярність швидкими темпами. Але це зовсім не означає, що реляційні БД стають рудиментом та не будуть використовуватись взагалі. Цей тип БД має свою нішу серед БД, він і досі активно використовується у багатьох додатках. Для різних потреб тепер використовуються різні типи БД. Архітектори програмного забезпечення мають можливість обирати сховище БД виходячи з природи самих даних, з того, яким чином ми хочемо цими даними маніпулювати та результуючих об'ємів даних, що будуть зберігатись.

Список використаних джерел:

1. NOSQL Databases [Електронний ресурс] – Режим доступу: http://nosql-database.org/ Перевірено: 5.06.2016.

2. NoSQL Databases Explained [Електронний ресурс] – Режим доступу: https://www.mongodb.com/ nosql-explained – Перевірено: 5.06.2016.

3. Advantages and Disadvantages of NoSQL databases – what you should know [Електронний ресурс] – Режим доступу: http://www.hadoop360.com/blog/advantages-and-disadvantages-of-nosql-databases-what-you-should-k – Перевірено: 5.06.2016.

Кузянін О.С.

студент,
Національний технічний університет України
«Київський політехнічний інститут»

MQTT – ПРОТОКОЛ ТЕХНОЛОГІЇ «ІНТЕРНЕТ РЕЧЕЙ»

Інтернет речей (IoT) – це технологія майбутнього. У сучасному світі все більше і більше пристроїв і датчиків, які збирають інформацію та обмінюються даними.

«Інтернет речей» дозволить їм встановлювати між собою зв'язок і забезпечить створення, в повному розумінні слова, розподілених додатків з між машинним (machine-to-machine – M2M) зв'язком. Він в 50 разів перевищить за обсягом традиційний Інтернет і забезпечить з'єднання в 10 разів більшої кількості пристроїв, ніж це дозволила свого часу зробити мобільна революція, і стане стимулом подальших перетворень нашого життя.