

Авторський колектив

Устенко С. В., д.е.н., проф., Рамазанов С. К., д.е.н., д.т.н., проф., Степаненко О. П., д.е.н., проф., Іванченко Г. Ф., к.т.н., проф., Сендзюк М. А., к.е.н., проф., **Ситник Н. В.**, к.е.н., проф., Шевченко К. Л., д.т.н., проф., Рінна С. П. д.е.н., проф., Тишков Б. О., к.е.н., доц., Галузинський Г. П., к.т.н., доц., Городній О. В., к.т.н., доц., Денісова О. О., к.е.н., доц., Гордієнко І. В., к.е.н., доц., Краснюк М. Т., к.е.н., доц., Курков М. С., к.е.н., доц., докторант, Помазун О. М., к.е.н., доц., Зінов'єва І. С., к.е.н., доц., Держук О. В., к.е.н., доц., Кривошеєв К. В., к.т.н., доц., Бібко О. О., асис., Кустаровський О. Д., асп., Рінна М. Б., к.е.н., доц., Гіваргізов І. Г., асп., Погореловська І. Д., к.е.н., доц., Бадер Омар Ахмад Далайін, к.е.н. (Йорданія)

Рецензенти

Дивак М. П. докт.техн.наук, проф.

Тернопільський національний економічний університет

Горбовий А. Ю. докт.техн.наук, проф.

Інститут інформаційних технологій Державної фіскальної Служби України

Джаладова І. А. докт.фіз.-мат.наук, проф.,

Київський національний економічний університет імені Вадима Гетьмана

Рекомендовано до друку Вченою радою КНЕУ

Протокол № 10 від 30.05.2019

I-74 **Інформаційні управляючі системи та технології /** За заг. ред. докт. екон. наук, професора Устенко С. В. — Київ : КНЕУ, 2019. — 419 с., [5] с.
ISBN 978-966-926-307-0

У монографії розглянуто концепції та методології проектування та розвитку інформаційних управляючих систем на підприємствах, у наукових інститутах, ІТ-сфері, виробництві, логістиці, освітньому процесі; методи, моделі та алгоритми вирішення завдань управління високотехнологічних виробничих систем, систем еволюційного прогнозування синергетичного ефекту злиття та поглинання підприємств, інтелектуальних систем прийняття рішень; принципи та стратегії проектування інформаційних систем з використанням технологій криптоекономіки та блокчейн, інформаційно-комунікаційних технологій підтримки інформаційної безпеки систем, створення гібридних інтелектуальних систем; упровадження технічних аспектів підвищення надійності пам'яті, методів отримання вимірювальної інформації, розроблення спеціалізованих систем цифрової обробки інформації.

Для викладачів, наукових працівників, аспірантів, студентів, спеціалістів, які займаються проблемами впровадження сучасних технологій при проектуванні та впровадженні інформаційних систем в управлінні підприємствами, організаціями, а також в ІТ-бізнесі, освіті.

УДК 004.9:33:62

*Розповсюджувати та тиражувати
без офіційного дозволу КНЕУ заборонено*

ISBN 978-966-926-307-0

© Колектив авторів, 2019
© КНЕУ, 2019

ІНТЕРАКТИВНА ПРОЦЕДУРА БАГАТОКРИТЕРІАЛЬНОЇ ОПТИМІЗАЦІЇ <i>Галузинський Г. П., к.т.н., доц.</i>	213
СТАН І ПЕРСПЕКТИВИ ПІДВИЩЕННЯ НАДІЙНОСТІ ПАМ'ЯТІ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ <i>Городний О. В., к.т.н., доц.</i>	223
МЕТОДИ ОТРИМАННЯ ІНФОРМАЦІЇ ДЛЯ УПРАВЛЯЮЧИХ СИСТЕМ І ПІДВИЩЕННЯ ЇЇ ДОСТОВІРНОСТІ <i>Шевченко К. Л., д.т.н., проф.</i>	254

РОЗДІЛ 3. ПЕРСПЕКТИВИ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

ІНТЕЛЕКТУАЛЬНЕ УПРАВЛІННЯ ТЕХНОГЕННИМ ПРОМИСЛОВИМ ОБ'ЄКТОМ В УМОВАХ ЗМІШАНОЇ НЕВИЗНАЧЕНОСТІ <i>Рамазанов С. К., д.т.н., д.е.н., проф.</i>	271
ІСО ЯК ІНВЕСТИЦІЙНИЙ ВИМІР КРИПТО-ЕКОНОМІКИ <i>Ріппа С. П., д.е.н., проф., Ріппа М. Б., к.е.н., доц. каф. фінансів УДФСУ (Ірпінь)</i>	293
РОЗРОБКА МЕТОДОЛОГІЇ ЕФЕКТИВНОГО ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ BIG DATA І DATA MINING ЯК ВАЖЛИВОЇ АНТИКРИЗОВОЇ СКЛАДОВОЇ РОЗРОБЛЕНОЇ КОМПЛЕКСНОЇ ПОЛІТИКИ ОПТИМІЗАЦІЇ ЛОГІСТИЧНОГО БІЗНЕСУ <i>Краснюк М. Т., к.е.н., доц. Кустаровський О. Д., асп.</i>	316
СИСТЕМИ ТА ТЕХНОЛОГІЇ ЦИФРОВОЇ ОБРОБКИ ІНФОРМАЦІЇ НА ОСНОВІ АЛГОРИТМІВ ПЕРЕТВОРЕННЯ ФУР'Є <i>Устенко С. В., д.е.н., проф., Тішков Б. О., к.е.н., доц., Погореловська І. Д., к.е.н, доц.</i>	337
СУЧАСНІ НАПРЯМИ РОЗВИТКУ КОНЦЕПЦІЇ БАЗ ДАНИХ <i>Ситник Н. В., к.е.н., проф.</i>	353
ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ БЛОКЧЕЙНУ <i>Гордієнко І. В., к.е.н., доц.</i>	376
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ ПІДТРИМКИ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ БАНКІВСЬКОЇ ДІЯЛЬНОСТІ <i>Гіваргізов І. Г., асп.</i>	404

СУЧАСНІ НАПРЯМИ РОЗВИТКУ КОНЦЕПЦІЇ БАЗ ДАНИХ

Бази даних — це невід’ємна складова будь-якої автоматизованої інформаційно-управляючої системи. Правильність побудови та якість бази даних — головні фактори, що впливають на ефективність і життєвий цикл інформаційно-управляючої системи.

Донедавна переважна більшість задач організаційно-економічного управління на рівні підприємств, установ і організацій вирішувалась з використанням реляційних баз даних. Реляційні бази даних більш як тридцять років (з 1980 року) домінували в ІТ-індустрії. Сутність побудови реляційних моделей найбільш повно викладено в роботах Дж. Мартіна [7], К. Дж. Дейта [3], Т. Конноллі [5] та інш.

Основою реляційних моделей є математичне поняття відношення, фізичним представленням якого є двовимірна плоска таблиця, яка складається з рядків (кортежів) і стовпчиків (атрибутів). Кожне відношення характеризує певну сутність предметної області. Реляційні бази даних (БД) — це чітко структуровані дані, що характеризують певну предметну область і представляються у вигляді пов’язаних між собою поіменованих двовимірних плоских таблиць, що перебувають під управлінням системи керування базами даних (СКБД). Реляційні БД мають такі переваги:

- чітке теоретичне підґрунтя, основане на реляційній алгебрі та теорії нормалізації реляційних відношень;
- мова запитів SQL (Structured Query Language), яка незважаючи на наявність синтаксичних відмінностей у різних системах керування базами даних, заснована на одному стандарті ISO і дуже зручна для реалізації запитів;
- наявність діаграми (схеми) бази даних, яка пов’язує між собою таблиці логічними зв’язками та забезпечує підтримку узгодженості та цілісності даних;
- можливість паралельного оброблення даних завдяки механізму підтримки транзакцій;
- інтегрованість, тобто можливість доступу до БД великої кількості користувачів.

Розглянемо основні проблемами використання реляційних БД.

Проблеми опрацювання надвеликих обсягів інформації. В інформаційних технологіях з’явився термін «великі дані» (Big Data), який

охоплює серію підходів, інструментів і методів оброблення структурованих і неструктурованих надвеликих об'ємів даних. Лише в одному Twitter формується близько 100 млн коротких повідомлень за добу. Компанія Google обробляє більше одного петабайта за годину. Спеціалісти з досвідної компанії IDC прогнозують, що до 2020 року, обсяги інформації збільшаться до 40 зеттабайт. Прогнозується, що обсяги даних збільшуватимуться вдвічі кожні два роки. У прес-релізі одного зі світових лідерів в ІТ-сфері компанії DELL EMC стверджується, що до 2020 року, цифровий всесвіт досягне обсягу в 40 зеттабайт, і до 2020 року в 15 разів зростуть обсяги інформації, яка генеруватиметься автоматично [11].

При опрацюванні Big Data потрібно враховувати їх характерні ознаками, які дістали назву «три V», а саме: фізичний об'єм (volume), швидкість (velocity), як зростання обсягів, так і швидкість їх оброблення та отримання необхідних результатів, різноманітність (variety), тобто можливість одночасного оброблення структурованих і напівструктурованих даних. Невпинно зростаючі обсяги даних, які обчислюються терабайтами й петабайтами, є перепорою для використання реляційних СКБД, які є мало продуктивними в таких випадках, а іноді не в змозі впоратися з великими навантаженнями.

Потреба підтримки агрегатів. Реляційні моделі підтримують лише атомарні атрибути й не підтримують їх агрегати, тому іноді їх називають безагрегатними (aggregate-ignore). Під агрегатом розуміється сукупність атрибутів, які інтерпретуються СКБД як єдине ціле. У таблицях реляційної бази дані зберігаються у вигляді атомарних атрибутів. Крім того, при нормалізації дані певного первинного документа розпорошуються на окремі атрибути, що зберігаються в різних реляційних відношеннях. У реальній дійсності дуже часто виникає потреба підтримки агрегатних даних, наприклад певної сукупності атрибутів чи окремих документів як єдиного цілого.

Проблема масштабування БД (Database Scalability). Щоб справлятися зі зростанням обсягів інформації та трафіку, використовують вертикальне та горизонтальне масштабування. Вертикальне масштабування (Vertical scaling) — це збільшення потужності комп'ютерів і серверів БД, що є доволі вартісним і має певну суто технічну межу. Горизонтальне масштабування (Horizontal scaling) у контексті баз даних — це збільшення кількості серверів, що паралельно виконують одну й ту саму функцію, тобто автоматичне розподілення даних між кількома серверами та підтримка кількох датацентрів з можливістю підключення нових комп'ютерів,

які об'єднуються в кластери. За рахунок цього кластери можуть використовувати просте апаратне забезпечення, вартість горизонтального масштабування є набагато меншою за вертикальне. Крім того, кластери є більш надійними, у разі виходу з ладу одного з комп'ютерів кластер продовжує функціонувати.

Основними способами горизонтального масштабування є реплікація та фрагментація (sharding) шардінг [8, 17, 18].

Реплікація — це підтримка ідентичних копій (реплік) на різних вузлах (серверах) розподіленої бази даних. Репліка може включати всю базу даних (повна репліка) чи певні її фрагменти. Підтримка реплікацій — внесення змін до розподіленої бази даних з метою підтримки ідентичності реплік.

Фрагментація, або шардінг — це розподіл бази даних на окремі складові для їх розміщення на різних вузлах (серверах). У розподілених базах даних можна використовувати одночасно обидві стратегії розподілу, тобто реплікацію можна поєднувати із шардінгом.

Реляційні СКБД не дуже підходять для кластеризації. Такі СКБД, як Oracle і Microsoft SQL Server, побудовані на концепції розподілу дискових підсистем і використовують кластерподібну файлову систему. Проте процедура фрагментації бази даних повинна управлятися з додатку, що контролює, який сервер і з якою порцією даних працює. При цьому втрачається можливість управління запитами, транзакціями, цілісністю даних і узгодженістю між складовими кластера. Крім того, перепорою застосування реляційних СКБД є вартість ліцензії. Адже вартість ліцензії комерційної СКБД при переході на роботу з кластерами різко зростає. Це спонукало до пошуку альтернатив реляційним базам даних і сприяло розвитку баз даних NoSQL. Зокрема великий вплив на розвиток нового напрямку баз даних зробили дві компанії Google і Amazon, які накопили великі обсяги інформації та були лідерами в горизонтальному масштабуванні й роботі з кластерами.

Значні витрати часу на адміністрування. Низка завдань, пов'язаних з масштабуванням і реплікацією, є доволі складними й трудомісткими в середовищі реляційних СКБД. У NoSQL-базах аналогічні операції займають лічені хвилини. Завдання встановлення та налаштування використання NoSQL-рішень зазвичай істотно простіші та менш трудомісткі, ніж у випадку з реляційними СКБД.

Домінуючи на ринку IT-технологій, реляційні бази даних довго були єдиним оптимальним вирішенням практично будь-якої задачі. Проте обмеження застосування реляційних баз даних спонукало до розвитку нового напрямку баз даних, що дістали назву NoSQL. NoSQL — це спільна назва всіх СКБД, що не мають у

своєму арсеналі мову запитів SQL. Ця назва не заперечує реляційні СКБД, вона трактується як «Not Only SQL», тобто «не лише SQL». Поява та розвиток нереляційних баз даних зумовлені швидким зростанням обсягів інформації та поширенням Інтернету. Інтернет-сервіси, такі як електронна пошта, соціальні мережі, блоги, електронна комерція, картографічні систем та інші, які надто часто оперують з дуже великими обсягами розподіленої неструктурованої чи слабо структурованої інформації, створили передумови для появи БД NoSQL.

Для переважної більшості WEB-додатків, таких як соціальні мережі, торговельні портали та різноманітні сайти, характерним є надвеликі обсяги даних і безліч користувачів, які можуть не лише переглядати, а й часто записувати чи змінювати дані.

Назвемо особливості баз даних NoSQL [2, 13, 17, 32]:

- не підтримується реляційна модель бази даних, тобто табличне подання даних;
- не використовується мова запитів SQL. Відсутність SQL в арсеналі баз NoSQL не є свідченням відсутності мови запитів у цих системах. Усі вони мають мову запитів, яка є відмінною від мови SQL. Проте деякі NoSQL-системи мають мову запитів, яка є дуже подібною до мови SQL;
- відсутність чітко заздалегіть заданої, регламентованої структури даних. Тобто в базах даних NoSQL можна за потреби додавати нові поля чи змінювати їх формати, для цього достатньо відобразити ці зміни в коді програми;
- відсутність діаграми (схеми) бази даних. Тобто NoSQL СКБД не підтримують статичну, жорстко задану схему даних, але при цьому мають можливість підтримки гнучких динамічних схем;
- можливість підтримки агрегатів і ненормалізованих даних;
- відсутність механізму підтримки транзакцій. Проте ця характеристика властива не всім базам даних NoSQL, зокрема, графові моделі мають механізм підтримки транзакцій;
- підтримка горизонтального масштабування на кластерах. Переважній більшості NoSQL-систем характерна ця властивість, тобто орієнтація на роботу з кластерами;
- робота з Big Data, тобто надвеликими обсягами даних;
- СКБД NoSQL — це в переважній більшості проекти з відкритим кодом.

Перелічені вище особливості є спільними характеристиками для баз даних NoSQL, проте жодна з них не є визначальною, у цьому і є проблема формулювання загального визначення для цього класу СКБД. Термін NoSQL уперше застосував у 1998 році

Карло Строщі (Carlo Strozzi) для своєї опенсорсної бази даних, у якій була відсутня мова запитів SQL. Проте тоді цей термін не мав нічого спільного з теперішніми базами NoSQL [23].

Пізніше, в 2009 році, на конференції у Сан-Франциско, на якій обговорювались питання ІТ-технологій в сфері зберігання й оброблення даних, Ерік Еванс для лаконічності запропонував термін NoSQL, який не мав під собою глибокого семантичного змісту [32]. Але цей термін прижився і став назвою нового напрямку розвитку концепції нереляційних баз даних. Незважаючи на те, що цей термін широко використовується, переважна більшість спеціалістів висловлює думку проте, що розвиток і поява нових баз даних NoSQL усе більш унеможливорює отримання ними спільного визначення, та стверджують, що аббревіатура NoSQL — це випадковий неологізм.

На сьогодні БД NoSQL дуже стрімко розвиваються та їх кількість постійно збільшується. Перелік найпоширеніших систем наведено в [33].

Про розвиток баз даних NoSQL свідчить рейтинг DB-Engines. Отже, згідно з рейтингом інформаційного видання DB-Engines, яке проаналізувало 347 СКБД, лідирують реляційні СКБД Oracle, MySQL і Microsoft SQL Server (рис. 1).

347 systems in ranking, May 2019

Rank			DBMS	Database Model	Score		
May 2019	Apr 2019	May 2018			May 2019	Apr 2019	May 2018
1.	1.	1.	Oracle	Relational, Multi-model	1285.55	+5.61	-4.87
2.	2.	2.	MySQL	Relational, Multi-model	1218.96	+3.82	-4.38
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1072.19	+12.23	-13.66
4.	4.	4.	PostgreSQL	Relational, Multi-model	478.89	+0.17	+77.99
5.	5.	5.	MongoDB	Document	408.07	+6.10	+65.96
6.	6.	6.	IBM Db2	Relational, Multi-model	174.44	-1.61	-11.17
7.	8.	9.	Elasticsearch	Search engine, Multi-model	148.62	+2.62	+18.18
8.	7.	7.	Redis	Key-value, Multi-model	148.40	+2.03	+13.06
9.	9.	8.	Microsoft Access	Relational	143.78	-0.87	+10.67
10.	11.	10.	Cassandra	Wide column	125.72	+2.11	+7.89

Рис. 1. Перша десятка СКБД згідно з рейтингом видання DB-Engines

Проте в десятці лідерів з’явилися системи NoSQL, які нарошують свою популярність, зокрема це СКБД MongoDB (5-е місце в рейтингу) і СКБД Elasticsearch, Redis, Cassandra, які посідають 7-е, 8-е і 10-е місця відповідно [11]. Усього в першій півсотні рейтингу СКБД налічується 20 систем NoSQL, що свідчить про їх стрімкий розвиток і популярність.

Теоретичним підґрунтям баз даних NoSQL, які в основному використовуються у розподілених системах, є теорема CAP

(Consistency, Availability, Partition tolerance), яка була запропонована Єріком Брюером (Eric Brewer), а також відома спеціалістам як гіпотеза Брюера [24].

У теоремі CAP зазначається, що розподілена система (це також стосується розподілених БД) не може одночасно мати більше двох з трьох таких характеристик: доступність (Availability), узгодженість (Consistency) і стійкість до розподілу (Partition tolerance).

Доступність (Availability) — полягає в тому, що будь-який запит до розподіленої бази даних отримує відгук навіть у разі виходу з ладу деяких вузлів, позаяк вузли, що залишились неушкодженими, продовжують функціонувати.

Узгодженість (Consistency) — дані в різних вузлах не суперечать один одному, і після виконання кожної операції система знаходиться в узгодженому стані, тобто всі користувачі в будь-який момент часу бачать однакові дані.

Стійкість до розподілу (Partition Tolerance) — система зберігає працездатність при її фізичному розподілі, тобто, якщо за певних збоїв у мережі система розпадається на непов'язані між собою групи вузлів, то, незважаючи на це, кожна така група продовжує функціонувати.

Виходячи з теореми CAP, при побудові розподіленої бази даних залежно від завдань, які потрібно вирішувати, обирається певний клас СКБД. Згідно з теоремою CAP усі СКБД поділяються на три класи (рис. 2): клас CA, які забезпечують узгодженість і доступність, клас CP підтримують узгодженість і стійкість до розподілу та клас AP, що характеризується доступністю і стійкістю до розподілу [21, 31].

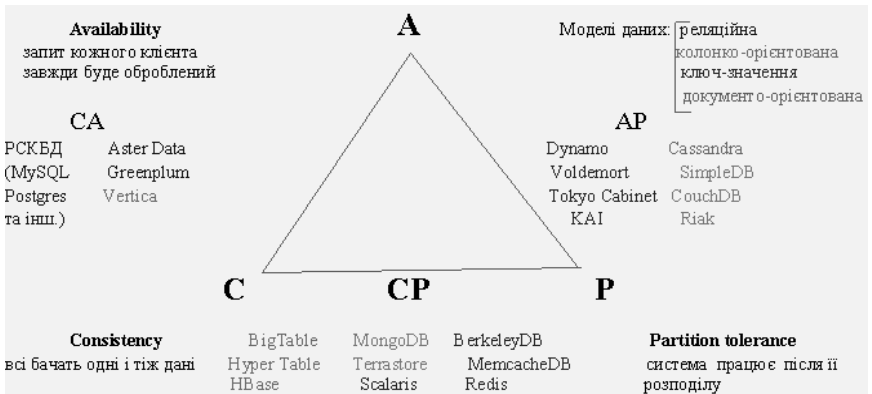


Рис. 2. Розбивка на класи деяких СКБД згідно теореми CAP

До першого класу належать традиційні реляційні СКБД, що неможливе їх використання в розподілених системах, у яких іноді потрібно нехтувати узгодженістю чи доступністю даних. Переважна більшість баз даних NoSQL належать до CP чи AP класу. Виходячи з цього, для баз даних NoSQL не потрібно чітко дотримуватись вимог механізму підтримки транзакцій — ACID (Atomicity, Consistency, Isolation and Durability). Абревіатура ACID вперше з'явилась у 1983 році в статті Тео Хаєрдера (Theo Haerder) і Андреаса Рейтера (Andreas Reuter) «Принципи відновлення баз даних, орієнтованих на операції» [37]. Транзакції характеризуються такими властивостями: атомарність (Atomicity), узгодженість (Consistency), ізольованість (Isolation), надійність (Durability).

Атомарність — трактується як «усе або нічого», тобто успішно виконана транзакція вносить зміни до бази даних, у протилежному випадку зміни не відбуваються.

Узгодженість — це властивість, яка означає, що транзакція буде успішно завершена лише тоді, коли зміни не порушують цілісності бази даних.

Ізольованість — означає локальне виконання кожної окремої транзакції, тобто транзакції, що виконуються паралельно, не впливають одна на одну.

Надійність — означає, що зміни, внесені до бази даних після успішного завершення транзакції, гарантовано зберігаються, навіть у разі збоїв обладнання чи програмного забезпечення.

Проте підтримка ACID-вимог в умовах розподілених систем збільшує складність таких систем і час реакції через велику кількість взаємодій у розподіленій мережі. Тому в базах даних NoSQL послаблені вимоги узгодженості на користь швидкості й доступності даних і їх горизонтального масштабування.

ACID-вимоги для баз даних NoSQL замінені BASE-вимогами, які є менш суворі до узгодженості, допускаючи тимчасову відмінність одних і тих же копій даних, але при цьому за певних умов збільшується доступність до розподіленої системи. BASE-вимоги формулюються як базова доступність (Basic Availability), яка гарантує завершення кожного запиту; гнучкий стан (Soft state), тобто стан системи може змінюватись незалежно від того, були введені нові дані чи ні; можлива узгодженість (Eventual consistency). При цьому слід зауважити, що відсутність підтримки вимог ACID стосується лише агрегатно-орієнтованих моделей баз даних, графові та інші безагрегатні

моделі підтримують ACID вимоги, подібно до того, як це роблять реляційні моделі баз даних.

Назвемо найпоширеніші моделі представлення даних у системах NoSQL [4, 13, 14, 17, 22, 28].

Модель — ключ/значення (Key-Value Stores) — агрегатно-орієнтована модель бази даних. Це проста форма організації даних у вигляді хеш-таблиць, яка характеризується високою швидкістю та дає змогу зберігати в пам'яті за певним ключем будь-які дані: числові, текстові чи серіалізовані об'єкти. Це максимально спрощена база даних, де все зведено до пари: ключ та самі дані. Доступ до такої бази даних дуже простий — це команди на зразок: `get` (найти дані за ключем), `set` (записати дані з ключем), `delete` (вилучити ключ і відповідні йому дані), `update` (оновити за ключем існуючі дані). Вторинні ключі в таких системах не використовуються. Важливим у такій базі даних є структура ключа. Ключі можна генерувати за допомогою певного алгоритму, чи використовувати як ключ ідентифікатори користувачів або їх електронні адреси, ідентифікатори веб-сесії та ін.

Модель ключ-значення має певні обмеження в її використанні. А саме:

- відсутня можливість встановлення відношень і зв'язків між наборами значень бази даних, що зберігаються з різними ключами;

- не можливо реалізувати пошук конкретних даних за відповідними їх значеннями, можлива лише вибірка всіх документів, що зберігаються за певним ключем.

- операції в базі даних «ключ-значення» в кожний конкретний момент часу обмежені одним ключем і не можливо одночасно виконувати операції з кількома ключами.

Представниками систем, у яких дані зберігаються за цим принципом, є: Redis, Riak, Scalaris, Tokyo Cabinet, Project Voldemort, Memcached, Berkeley DB, HamsterDB, Amazon DynamoDB, Oracle NoSQL Database.

Як приклад моделі ключ/значення розглянемо систему Redis (Remote Dictionary Service — віддалена служба словників) [29], яка згідно з рейтингом СКБД (див. рис. 1) знаходиться на восьмому місці. СКБД Redis — це система на мові C, з відкритим кодом, яка підтримує рядки (Strings), множини й списки даних (Lists) та інші структури. Перша версія Redis вийшла в 2009 році. На сьогодні Redis використовується в таких великих проєктах, як Twitter, Instagram, Digg, Github, StackOverflow, Flickr та ін.

Окрім операцій пошуку за ключем, запису за ключем, вилучення та оновлення за ключем Redis підтримує горизонтальне масштабування та асинхронні реплікації на зразок «головний-підпорядкований». Клієнтські бібліотеки для роботи з Redis доступні для переважної більшості мов програмування.

У Redis основним при роботі з даними є ключ. Ключ — це поле, що ідентифікує фрагменти даних, тобто їх значення. У Redis усі доступні операції виконуються лише з ключем, система не надає можливості маніпулювати з конкретними значеннями даних, що зберігаються в БД.

Виходячи з наданих характеристик, можна зробити висновок, що СКБД Redis є специфічною системою, яка не підходить для будь-якого сценарію роботи з даними. Redis найчастіше застосовується як сховище ключів для даних, які важко й довго обчислювати щоразу заново. У ключах можна задавати термін зберігання даних. Redis дуже ефективно застосовується в системах багатостороннього зберігання даних, у яких використовується одночасно кілька СКБД. У таких системах на Redis покладається завдання швидкісного заповнення бази даних.

Колонко-орієнтована модель (Tabular Column-oriented) — агрегатно-орієнтована модель бази даних, у якій агрегат містить дані одного чи кількох стовпчиків таблиці. У цих моделях підхід до структуризації даних відмінний від реляційних систем, у яких одиницею зберігання, пошуку та маніпулювання даними є кортежі (рядки) таблиць. Такий підхід використовуються для тих задач, у яких потрібно зчитувати по кілька стовпчиків з кількох структур одночасно. Тоді краще одиницею зберігання в базі даних зробити групи стовпчиків. Це зручно при обробленні архівних даних і каталогів, у яких пошук і вибірки виконуються за стовпчиками, а не за рядками. Бази даних цього типу організують стовпчики в сімейства (асоціативні масиви). Кожен стовпчик повинен бути частиною певного сімейства й одиницею доступу, при цьому дані конкретного сімейства можуть бути одночасно доступним. Представниками таких колонко-орієнтованих систем зберігання даних є: BigTable, HyperTable, Hypertable, HBase, Hadoop, Cassandra, Amazon SimpleDB.

Як приклад СКБД, в основу якої покладено *колонко-орієнтовану модель*, розглянемо дуже поширену на практиці СКБД Cassandra. Cassandra є проектом з відкритим кодом на мові Java. Система Cassandra [19] була розроблена й використовувалась у Facebook. В її основу взято ідеї Google BigTable і Amazon Dynamo. У 2009 році код Cassandra було передано фонду

Apache і на сьогодні ця СКБД використовується в таких компаніях: Adobe, CERN, Cisco, IBM, HP, Comcast, Disney, eBay, Netflix, Sony, Rackspace, Reddit, Digg и Twitter.

Модель даних Cassandra складається з стовпчиків, рядків сімейства стовпчиків і простору ключів. Найменшою структурною одиницею зберігання даних у СКБД Cassandra є стовпчик (*Column*). Стовпчик — це кортеж (*tuple*), що складається з ім'я (*name*), значення (*value*) і часової мітки (*timestamp*). Ім'я стовпчика є його ключем. За часовим параметром можна визначити термін зберігання даних. Усі перелічені три параметри задаються клієнтом, зокрема й *timestamp*. Таким чином, часові параметри в клієнтів повинні бути синхронізованими, позаяк *timestamp* використовується для вирішення конфліктів при внесенні змін до даних.

Стовпчики можуть об'єднуватись у сімейства. Сімейство стовпчиків (*column family*) — це рядок, який містить дані певної кількості стовпчиків, що ідентифікуються ключем рядка. У сімейство об'єднуються семантично споріднені стовпчики, які потрібно шукати чи считувати як єдине ціле. Рядок ідентифікується ключем (*row key*) і являє собою ключ і пов'язані з ним набори стовпчиків. У Cassandra, кожне сімейство стовпчиків зберігається в окремому файлі, який відсортований за ключем рядка. Сімейство стовпчиків можна порівнювати з рядками в реляційній таблиці, у якій ключ ідентифікує рядок, а рядок складається з множини атрибутів. Відмінність від реляційної бази полягає в тому, що різні рядки одного сімейства стовпчиків можуть мати різну структуру, тобто вміщувати різні стовпчики. Крім того, за потреби певні рядки, а не всі разом можуть бути поповнені новими стовпчиками.

Якщо сімейство стовпчиків складається з звичайних стовпчиків, то воно називається стандартним (*standard column family*). Якщо стовпчик являє собою асоціативний масив стовпчиків, то тоді він є суперстовпчиком (*super column*). Суперстовпчик — це стовпчик-контейнер, що містить у собі інші стовпчики. Тобто суперстовпчик — це група стовпчиків під одним ключем. Простір ключів (*Keyspace*) — це група з певної кількості логічно пов'язаних сімейств стовпчиків. Простір ключів подібний до діаграми (схеми) у реляційній базі даних. Для реалізації запитів у СКБД Cassandra використовується мова CQL (*Cassandra Query Language*), яка за синтаксисом є дуже подібною до мови SQL. Для прискорення виконання запитів у системі можна створювати вторинні індекси.

У системі Cassandra зберігається лише остання версія даних, тобто, якщо в сімейство стовпчиків надходить стовпчик з ключем та іменем, які уже існують в базі, а мітка часу більша за існуючу, то на місце старих даних перезаписуються нові.

Cassandra характеризується великою швидкістю й легко масштабується. При горизонтальному масштабуванні дані розподіляються між кластерами, які не мають ведучого, тому операції читання та запису можна виконувати на будь-якому кластері. Система добре працює як на двох так, і на тисячі серверів. При цьому переконафігування при поповненні новими вузлами виконується повністю в автоматичному режимі. Крім легкого поповнення кластерів, виявлення збоїв і відновлення роботи системи також виконується автоматично. Реплікації можуть здійснюватись як у межах одного кластера, так і між територіально віддаленими кластерами.

Документно-орієнтована (Document-Oriented) модель — агрегатно-орієнтована база даних, у якій агрегат — це зазвичай окремий документ, який представлений у форматі, JSON. У таких базах кожний запис зберігається як окремий документ, що має свій власний набір атрибутів. Можна зберігати складні документи, тобто один батьківський документ може вмішувати кілька вкладених дочірніх документів. На відміну від баз даних «ключ-значення» в документно-орієнтованих базах даних є можливість реалізації запитів до певних даних конкретного документа, не викликаючи при цьому весь документ по його ключі. Поширеними системами, які підтримують документно-орієнтовану модель, є: CouchDB, MongoDB, OrientDB, RavenDB, Terrastore. У базах даних такого типу немає поняття схеми даних, уся інформація зберігається в форматі JSON (*JavaScript Object Notation*) чи його похідних. Наприклад, у системі MongoDB використовується формат BSON, який є розширенням JSON і дає змогу зберігати бінарні дані. Анонсована спроба створення специфікації спільної мови UnQL для баз даних такого типу.

Незважаючи на те, що на сьогодні відомо безліч баз даних NoSQL-систем, розглянемо MongoDB, яка посідає п'яте місце в рейтингу найбільш поширених СКБД [9, 16, 26, 30]. MongoDB — це система керування документно-орієнтованою, розподіленою базою даних, яка орієнтована на WEB-додатки та інфраструктуру Інтернету. Перша версія СКБД MongoDB розроблена в 2009 р. компанією 10gen. Продукт MongoDB реалізований на мові C++; він має драйвери для багатьох мов програмування, включаючи

C, C#, C++, Erlang, Haskell, Java, JavaScript, Perl, PHP, Python, Ruby и Scala. Крім того, передбачений інтерфейс командного рядка для JavaScript. MongoDB написана на C++, тому її можна використовувати на різних платформах. MongoDB може бути інстальована в середовищі Windows, Linux, MacOS, Solaris. Це система з відкритим кодом, яка є зручною при горизонтальному масштабуванні та роботі з кластерами й характеризується високою продуктивністю. У СКБД MongoDB реалізовано механізм горизонтального масштабування, який виконує розбивку даних на певні сегменти. Це механізм автошардування (auto-sharding), який автоматично керує розподілом даних між вузлами. Система сама додає нові вузли й автоматично переходить на резервний сервер при виникненні збою основного сервера. Шардінг у MongoDB забезпечує просте й зручне додавання нових машин без простоїв системи, допускається масштабування до тисячі вузлів і тим самим реалізується підвищення відмовостійкості. Реплікації у MongoDB — це механізм синхронізації даних між кількома копіями одного об'єкта. Записи посилаються на один основний сервер (master), який синхронізує свій стан з іншими серверами, які виступають як підпорядковані (slave). Реплікасет (множина реплік) — це форма асинхронної реплікації між основним і підпорядкованими вузлами, яка забезпечує автоматичну відмовостійкість і відновлення вузлів після збою. Реплікасет складається з двох чи більше вузлів, які є копіями один одного, при цьому всі члени реплікасета є рівнозначними, а основний вузол обирається автоматично.

У СКБД MongoDB відсутня схема (діаграма) бази даних та оператори JOIN, за допомогою яких у реляційних моделях здійснюється зв'язування таблиць між собою. Проте MongoDB також має зручну й прозору модель даних, яка підтримує різноманітні ієрархічно організовані структури даних, що дає змогу не використовувати численні зв'язування багатьох таблиць, як цього потребують реляційні бази даних. Так, у системі електронної комерції інформація про певний товар може розміщуватися в кількох таблицях реляційної бази даних, а в системі MongoDB переважна більшість інформації про певний товар можна представити в одному документі, вкладаючи один документ в інший. Для реалізації на зразок співвідношення «багато до багатьох» у MongoDB використовуються масиви, які також можуть бути представлені в документі. Наприклад, автомобілів може багато різних марок, тоді всі вони також описуються в документі, що характеризує певний документ як масив марок.

Таким чином, основою MongoDB є документ (document), який представляється як упорядкований набір ключів з певними значеннями. Документи об'єднуються в колекції (collection). Колекція — це сокупність структурно чи семантично подібних документів. Одна колекція може вміщувати документи різної структури з різними типами значень полів і ключів. Спільним для об'єднання документів в одну колекцію може бути предметна область, яку вони характеризують. Кожна колекція ідентифікується певним ім'ям, яке не повинно перевищувати 128 символів, завжди починається з букви чи цифри, також може вміщувати знак крапки. Документ обов'язково повинен мати первинний ключ, який має бути унікальним у межах колекції, що записується в полі `_id` певного документа. Первинний ключ може задаватися розробником чи генеруватися системою як ідентифікатор об'єкта. Якщо первинний ключ не заданий розробником, то СКБД автоматично створює 12-байтовий, 16-значний бінарний ідентифікатор. Структуру ідентифікатора об'єктів подано на рис. 3, вона вміщує чотирьох байтну мітку часу типу `timestamp`, трьох байтний ідентифікатор комп'ютера, 2-байтний ідентифікатор процесу і трьох байтне поле для лічильника. Така структура дає змогу створювати до 16 777 216 унікальних ідентифікаторів об'єктів (`ObjectId`) за секунду для одного процесу. Перші 9 байт гарантують унікальність комп'ютера, наступні 3 байта — унікальність упродовж секунди конкретного процесу.

Ідентифікатор об'єкта має таку структуру:

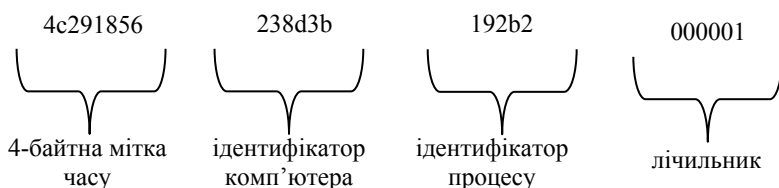


Рис. 3. Структура ідентифікатора об'єктів

Стандартна часова мітка дає можливість за необхідності встановити дату та час створення документа з точністю до секунди.

Еквівалентним поняттям колекція є таблиця реляційної бази даних. Колекція може містити певну кількість документів. База даних MongoDB — це набір колекцій (рис. 4).

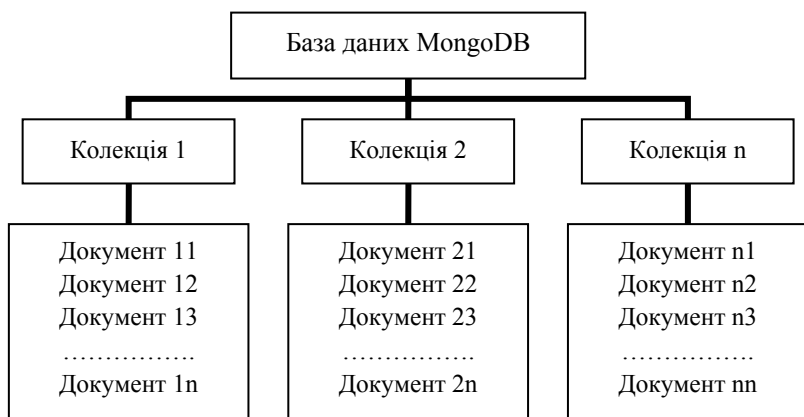


Рис. 4. Структурна схема бази даних MongoDB

Один примірник MongoDB може підтримувати кілька баз даних, кожна з яких незалежна.

Кожен документ характеризується певним набором полів, що можна розглядати як стовпчик таблиці, а окремий документ — як рядок таблиці. Колекція не має певної схеми, що дає змогу зберігати в рамках однієї колекції кілька різних документів.

На рис. 5 показано відповідність структур реляційної та документо-орієнтованої бази даних MongoDB.

Для опису складових бази даних використовується JSON-формат (JavaScript Object Notation), який дає можливість описувати складні структури документів. На фізичному рівні всі документи зберігаються на дисках у двійковому форматі BSON (БиСон) — скорочено від binary JSON. Документ являє собою набір пар: «ключ (key) — значення (value)». База даних і колекції створюються при вставках першого документа.

Нижче наведено приклад документу, який характеризує одного клієнта:

```
{«username»: «Anna»,
 «userssurname»: «Gurina»}
```

У виразі «username»: «Anna» — username — key, а Anna — значення. Для цього документа вкажемо колекцію з ім'ям users, де буде він збережений:

```
>db.users.insert ({«username»: «Anna»,
 «userssurname»: «Gurina»})
```

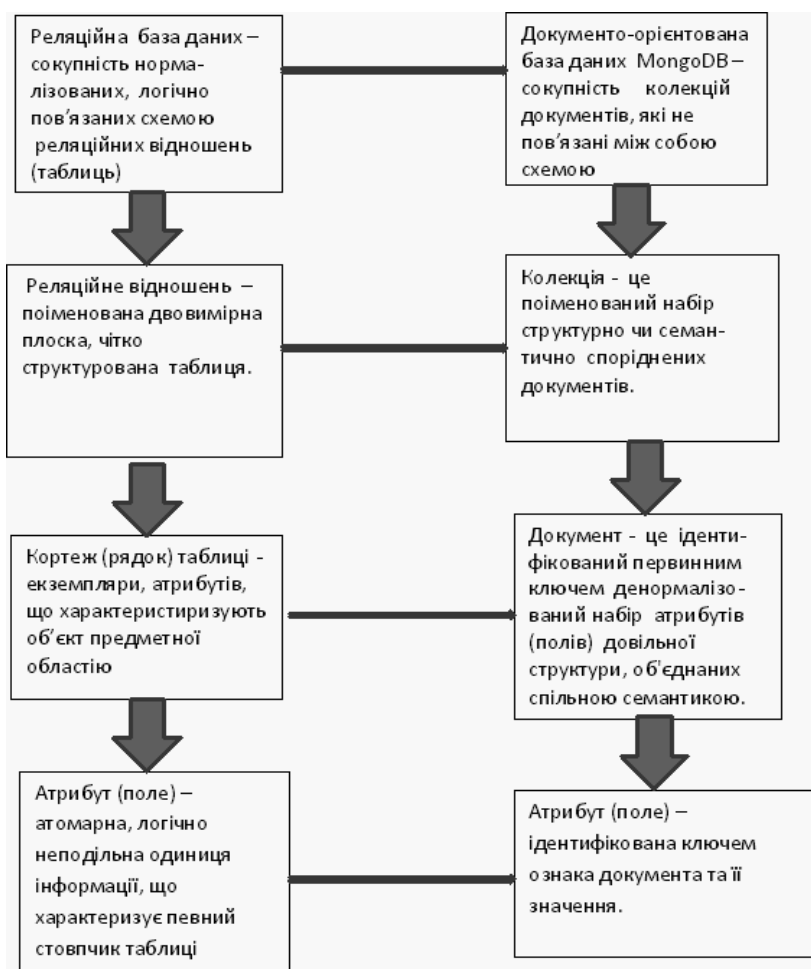


Рис. 5. Схема відповідності між структурними елементами реляційної та документно-орієнтованої бази даних MongoDB

У MongoDB можуть використовуватись різні типи даних. Назвемо їх.

- Integer — використовується для зберігання цілочисельних значень. Залежно від сервера може бути як 32-бітовим, так і 64-бітовим.
- Double — застосовується для зберігання значень з плаваючою крапкою.

- Boolean — використовується для зберігання логічних (true/false) значень.
- String — використовується для зберігання символьних рядків.
- Arrays — даний тип даних застосовується для зберігання масивів значень за одним ключем.
- Object — використовується для вбудованих документів.
- Symbol — використовується також, як і String, але зазвичай резервується для мов, які застосовується спеціальні символи.
- Null — використовується для зберігання значення Null.
- Timestamp — використовується для зберігання дати та часу.
- Min / Max — використовується для порівняння значень з найбільшим і найменшим BSON (Binary JSON) елементом.
- Object ID — використовується для зберігання ID-документа.
- Regular Expression — використовується для зберігання регулярних виразів.
- Code — використовується для зберігання JavaScript коду в документі.
- Binary data — даний тип даних дає змогу зберігати бінарні дані.
- Date — використовується для зберігання поточної дати або часу в UNIX-форматі.

Документи об'єднуються в колекції. Приклад колекцій бази даних Інтернет-магазину наведено на рис. 6.

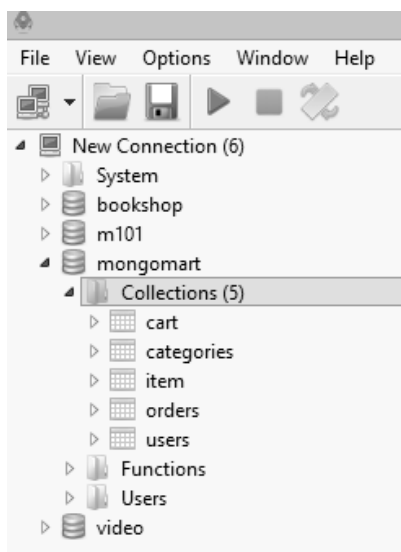


Рис. 6. Набір колекцій бази даних Інтернет-магазину

Наведений приклад бази даних Інтернет-магазину містить п'ять різних колекцій, а саме:

- Cart (Кошик);
- Categories (Категорії);
- Item (Товар);
- Orders (Замовлення);
- Users (Клієнт).

У MongoDB можна створювати індекси, які прискорюють пошук необхідних документів. Ключі індексації можуть бути простими чи складовими, тобто вміщувати ключі вкладених документів чи файлів.

СКБД MongoDB може зберігати різні документи з різними наборами даних, але розмір документа обмежується 16 мб. Для вирішення проблеми цього обмеження у MongoDB існує файлова система GridFS, яка дає змогу вийти за рамки обмеження, тобто зберігати дані більш як 16 мб.

GridFS вміщує дві колекції: колекцію files, яка зберігає імена файлів і метадані, колекцію chunks, у які зберігаються дані файлів, представлені у вигляді окремих сегментів розміром по 256 кб. До складу MongoDB входить спеціальна утиліта mongofiles, яка підтримує технологію GridFS.

MongoDB не має механізму підтримки транзакцій, але дає можливість за допомогою різних способів виконувати атомарне оновлення структур документів.

MongoDB рекомендують використовувати за великої кількості запитів, що одночасно надходять у систему (більше тисячі за секунду), а також за великої кількості операцій запису. Саме можливість швидкого запису є однією з головних переваг цієї бази даних.

Описані вище агрегатно-орієнтовані моделі інтерпретують агрегат як єдине ціле при роботі з даними. Атомарність підтримується лише на рівні вмісту певного агрегата, позаяк агрегатно-орієнтовані бази даних не підтримують вимог ACID. Якщо потрібно оновлювати кілька агрегатів одночасно, то прикладна програма повинна самостійно реагувати на виникнення збою при одночасному зверненні до декількох агрегатів. Таким чином, якщо для даних характерним є наявність великої кількості зв'язків, то краще використовувати графові моделі.

Графова модель (Graph Databases) — це безагрегатна модель бази даних. Поява агрегатно-орієнтованих моделей зумовлена потребою зберігання великих записів, які характеризуються відсутністю або простими зв'язками. Проте іноді потрібно зберігати

записи з великою кількістю зв'язків між ними. Для таких випадків підходять графові моделі баз даних. У таких моделях вершини й ребра графа використовуються для представлення даних. Ця модель якоюсь мірою розширює можливості моделі key/value, бо, крім даних, що зберігаються у вершинах графа є дуги, що вказують на зв'язки між ними. Графові моделі найчастіше працюють на одному сервері, а не розподілені по кластерах. Крім того, графові моделі підтримують вимоги ACID, тобто транзакції. Графові моделі відносять до баз даних NoSQL, бо вони не підтримують реляційну модель і не мають мову SQL. Дуги в графових моделях можуть бути двонаправленими, що дає змогу відображати складні взаємозв'язки між даними предметної області. Наприклад, вузли detail і virib можуть бути пов'язаними двонаправленою дугою, позаяк деталей може входити до багатьох виробів, а виріб складається з багатьох деталей, тобто графові моделі підтримують зв'язки між сутностями типу Б:Б.

Поширеними системами, що підтримують цей тип моделей, є: Neo4j, AllegroGraph, ActiveRDF, Infogrid і HyperGraphDB, Infinite Graph, Bigdata, OrientDB.

Ураховуючи те, що графові бази даних оперують зв'язаними вузлами, переважна більшість графових СКБД не працює з кластерами, тобто не має можливості розподілу вузлів між кількома серверами. Винятком є база даних Infinite Graph, яка підтримує розподіл вузлів по кластеру серверів.

Подальшим розвитком моделей, заснованих на графах, є представлення моделі бази даних у вигляді гіперграфа. Гіперграф — це граф, в якому одне ребро може мати більш як дві вершини. Система HyperGraphDB розширює концепцію гіперграфа й дає змогу ребру посилатися на інше ребро.

Графові бази даних знайшли успішне застосування в системах, для яких характерним є численні взаємозв'язки між даними. Прикладом можуть слугувати соціальні мережі, геоінформаційні, картографічні, логістичні системи та різні системи маршрутизації та диспетчирезації. Останнім часом графові бази даних почали використовувати в банківських і фінансових установах для попередження та запобігання шахрайських операцій з кредитами та картковими операціями. Графові СКБД можуть також бути використаними в урядових системах, які ведуть боротьбу зі злочинністю і запобігають тероризму.

Як приклад системи, побудованої на графовій моделі, розглянемо СКБД Neo4j [14, 25, 38]. СКБД Neo4j має зручний веб-інтерфейс з візуалізацією графів, підтримує транзакції. Будь-яка

операція зміни даних, крім операції читання, оформлюється як транзакція. База даних Neo4j написана на Java, тому її можна вмонтовувати в додатки, які написані на мові Java. При роботі на одному сервері дані завжди узгоджені. При роботі на кластері запис даних на ведучий сервер синхронізується з підпорядкованими вузлами й навпаки, запис даних на підпорядковані сервери синхронізується з ведучим сервером. Сервер, який першим приєднався до кластеру, стає ведучим вузлом. У разі виходу з ладу ведучого вузла, кластер обирає ведучий вузол серед йому доступних.

СКБД Neo4j підтримує декларативну мову запитів до графа Cypher, синтаксис якої дещо схожий на мову SQL. Ця мова дає змогу створювати, робити вибірки, оновлення та вилучення даних. Крім того, за допомогою мови Cypher можна проводити операції впорядкування, агрегування, пропуску та обмеження даних.

Основною моделі бази даних Neo4j є орієнтований граф, який вміщує такі елементи:

- вузли (nodes), що вміщують інформацію про певні сутності предметної області;
- поіменовані та орієнтовані дуги (relationships), які характеризують зв'язки між вузлами;
- вузол може мати кілька рекурсивних зв'язків;
- вузли мають властивості (properties), їх можна розглядати як документи, які характеризуються парами властивостей «ключ-значення»;
- вузли можуть бути поміченими однією чи кількома мітками (labels), які вказують на їх роль у наборі даних і за допомогою яких можна групувати вузли. Вузли, які помічені однією міткою, належать до одного набору даних.

Neo4j підтримує такі типи даних: булеві змінні (Bool), байт (byte), короткі цілі числа зі знаком (short), цілі числа зі знаком (Int), довгі числа зі знаком (long), числа з плаваючою комою (float), числа подвійної точності з плаваючою комою (double), символні типи й текстові ланцюжки. У СКБД Neo4j є функція для пошуку найкоротшого шляху, при цьому є можливість вибору одного з алгоритмів, які можуть вирішувати цю задачу: алгоритм Дейкстри, алгоритм А* (A-star) та ін.

Починаючи з версії Neo4j 1.3, система може підтримувати граф розміром до 32 мільярдів вузлів і 64 мільярдів властивостей. Потреба в масштабованих рішеннях, які надавали б переваги баз даних NoSQL, спонукали провідних розробників класичних реляційних баз даних до розробки нових продуктів, які підтри-

мують бази даних NoSQL. Компанія ORACLE розробила розподілену базу даних Oracle NoSQL Database 3.0 типу «ключ-значення», яка призначена для масштабування веб-додатків [10]. Компанія IBM DB2 має розробки, які підтримують сховище на основі NoSQL-графа [8].

Компанія Microsoft у версії СКБД SQL Server 2016 запропонувала низку функцій, що реалізують гібридний підхід, який дає змогу перетворювати реляційні таблиці в нереляційні об'єкти JSON СКБД Azure SQL Database, і навпаки [15]. Крім того, Microsoft розробила базу даних NoSQL DocumentDB [20].

Для можливості горизонтального масштабування без відмови від підтримки транзакцій і мови SQL використовується нова концепція побудови баз даних, яка дістала назву NewSQL. Ця назва була запропонована аналітиком 451 Group Метью Аслеттом (Matthew Aslett) у 2011 році для нового класу розподілених високопродуктивних систем, що масштабуються мають у своєму арсеналі мову SQL і підтримують ACID-транзакції [35]. Дуже часто такі системи ще називають Scalable SQL, чим підкреслюють їх відмінність від класичних односерверних реляційних СКБД.

Необхідність розробки таких баз даних зумовлена вимогами деяких OLTP-задач (Online Transaction Processing), які потребують обробки великих обсягів даних і масштабування, але для яких не може бути прийнятною слабка узгодженість даних NewSQL. Ці системи приблизно в 50 разів перевищують швидкість традиційних баз даних OLTP-систем [35].

NewSQL-системи поділяють на такі групи [35, 36]:

- 1) системи с новими архітектурами;
- 2) нові SQL-двигуни для зберігання даних MySQL;
- 3) прозорі кластери.

Системи с новими архітектурами є найбільш популярними, вони побудовані на нових платформах для зберігання даних і відразу створювались з урахування розподіленої архітектури й роботи з кластерами. Бази даних цього класу для підвищення швидкості оброблення даних використовують оперативну пам'ять чи нові типи дисків (флеш-пам'ять/SSD) для зберігання первинних даних. Системи, що використовують оперативну пам'ять для зберігання даних дістали назву In-memory СУБД [1]. In-memory-системи характеризуються значно більшою продуктивністю, позаяк процесор значно швидше працює з ОЗУ, ніж з дисковою пам'яттю. Появі БД, що зберігаються в оперативній пам'яті, сприяла тенденція до зниження вартості ОЗУ, що намітилась у середині 2000-х років, і розвиток багатоядерних процесорів.

Недостатність пам'яті ОЗУ одного комп'ютера компенсується об'єднанням їх у кластери. Оперативна пам'ять є енергозалежною, тому для запобігання втрати даних, що зберігаються в ОЗУ, СКБД з новою архітектурою підтримують ведення журналу операцій і з певною періодичністю копіюють базу на диски.

Прикладами таких систем є: Spanner, Clustrix, OrientDB, VoltDB, MemSQL, SQLFire і GemFire XD від Pivotal, SAP HANA, FoundationDB, NuoDB, TransLattice, ActorDB, Trafodion, CockroachDB та інші. Ці СКБД об'єднали в собі кращі сторони NoSQL-систем і класичних реляційних баз даних. Їх переваги — у вищій продуктивності та більшій швидкості відгуку, яка, за оцінками розробників, у кілька разів більша, ніж у традиційних розподілених СКБД.

Нові SQL-двигуни для зберігання даних MySQL. Для реалізації горизонтального масштабування MySQL розроблені набори движків зберігання, які включають Xeround, Akiban, кластер MySQL NDB, GenieDB, Tokutek та ін. Так, двигун MySQL Cluster розподіляє дані між множиною серверів, підтримує архітектуру shared nothing, а дані можуть знаходитись в оперативній пам'яті чи на дисках. У MySQL Cluster реалізований механізм синхронної реплікації, який забезпечує масштабування, але значно менших масштабах, ніж це можливо в системах NoSQL.

Прозорі кластери — це системи для масштабування поверх традиційних СКБД, які забезпечують проміжний шар (middleware), для автоматичного розподілу баз даних на кілька вузлів. При цьому база даних зберігається в своєму початковому форматі та забезпечує можливість прозорого її підключення до кластера, щоб забезпечити масштабованість. Приклади таких систем: ScalArc, Schooner MySQL, dbShards та ScaleBase (комерційні) і Continuent Tungsten (з відкритим кодом).

Продуктивність таких рішень менша, ніж систем з новими архітектурами, але перевагою їх є те, що вони не потребують модифікації існуючих раніше розроблених додатків.

Проаналізувавши основні тенденції у розвитку баз даних і СКБД, можна дійти висновку, що на сьогодні не існує універсальної, інтегрованої системи, що могла б задовольнити всім вимогам. Тому для OLTP-систем, які потребують підтримку транзакцій, підходять класичні реляційні СКБД для централізованого оброблення даних і системи NewSQL для оброблення надвеликих розподілених даних. Для зберігання та оброблення розподілених, неструктурованих WEB-орієнтованих даних найоптимальнішими є бази даних NoSQL.

Література

1. Базы данных в оперативной памяти // Википедия, свободная энциклопедия. — URL: https://en.wikipedia.org/wiki/In-memory_database
2. *Джигнеш Пател*. Операционные СУБД NoSQL: сегодня и завтра // Открытые системы — 2016. — № 3. <https://www.osp.ru/os/2016/03/13050248/>
3. *Дейт К. Дж.* Введение в системы баз данных: Пер. с англ. — 6-е изд — К.: Диалектика, 1998. — 784 с.
4. Інформаційні системи в економіці: Монографія / За заг. ред. С. В. Устенка. — К.: КНЕУ, 2012. — 425 с.
5. *Коннолли Т., Бегг К., Страчан А.* Базы данных: проектирование, реализация и сопровождение. Теория и практика. Учеб. Пособие. — 2-е изд.: Пер. с англ.: М.: Изд. дом «Вильямс», 2000. — 1120 с.
6. *Кузнецов С. Д., Посконин А. В.* Распределенные горизонтально масштабируемые решения для управления данными. http://www.ispras.ru/proceedings/docs/2013/24/isp_24_2013_327.pdf
7. *Мартин Дж.* Организация баз данных в вычислительных системах: Пер. с англ. — М.: Мир, 1980. — 662 с.
8. Обеспечение масштабируемости данных облачного уровня с помощью баз данных типа NoSQL <http://www.interface.ru/home.asp?artId=35468> (DB2)
9. Онлайн-руководство по MongoDB <https://metanit.com/nosql/mongodb/>
10. Oracle выпускает Oracle NoSQL Database 3.0 <https://www.oracle.com/ru/corporate/pressrelease/2-17538.html>
11. Прес-реліз компанії DELL EMC <https://ukraine.emc.com/about/news/press/2012/20121211-01.htm>
12. Рейтинг DB-Engines . <https://db-engines.com/en/ranking>
13. *Редмонд Уилсон*: Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL. — М.: ДМК-Пресс, 2017. — 384 с.
14. *Робинсон Ян, Вебер Джим, Эйфрем Эмиль*. Графовые базы данных: новые возможности для работы со связанными данными: Пер. с англ. — 2-е изд. Р. Н. Рагимова; Науч. ред. А. Н. Кисилев. — М.: ДМК Пресс, 2016. — 256 с.
15. Реляційний підхід до NoSQL <https://blogs.msdn.microsoft.com/sqlserverstorageengine/2015/09/01/combining-relational-and-nosql-concepts-in-sql-server/ON>.
16. Руководство по MongoDB (полная версия) <http://proselyte.net/tutorials/mongodb/>

17. Фаулер, Мартин, Садаладж, Прамодкумар Дж. NoSQL: новая методология разработки нереляционных баз данных: Пер. с англ. — М.: Изд. дом Вильямс, 2013. — 192 с.
18. Шардирование баз данных <https://ru.dmstu.wiki/> Шардирование баз данных
19. Apache Cassandra <http://cassandra.apache.org/>
20. DocumentDB: база данных NoSQL от Microsoft <http://habrahabr.ru/hjst/234233>
21. CAP Theorem within NoSQL <http://developerjam.blogspot.com/2013/09/cap-theorem-within-nosql.html>
22. C. Strauch .NoSQL Databases <http://www.christof-strauch.de/nosql dbs.pdf>
23. C. Strozzi. NoSQL: A Relational Database Management System http://www.strozzi.it/cgibin/CSA/tw7/I/en_US/nosql/Home%20Page
24. E. Brewer «Towards Robust Distributed Systems» в ACM Symposium on the Principles of Distributed Computing, Portland, Oregon, 2000. <http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf>
25. Graph Databases <https://neo4j.com/product/>
26. Karl Seguin. The Little MjogoDB Book <http://jsman.ru/mongo-book/index.html>
27. NewSQL — Википедия <https://ru.wikipedia.org/wiki/NewSQL>
28. R. Cattell Scalable SQL and NoSQL Data Stores <http://www.cattell.net/datastores/Datastores.pdf>
29. Redis <http://redis.io/documentation>
30. MongoDB Production Deployments: <http://www.mongodb.org/about/production-deployments/>
31. Nathan Hurst. Visual Guide to NoSQL Systems <http://blog.nahurst.com/visual-guide-to-nosql-systems>
32. NoSQL базы данных: понимаем суть <http://habrahabr.ru/post/152477>
33. List of NoSQL databases <http://nosql-database.org/>
34. Oracle выпускает Oracle NoSQL Database 3.0 <https://www.oracle.com/ru/corporate/pressrelease/2-17538.html>
35. P. Venkatesh NewSQL — The New Way to Handle Big Data <http://opensourceforu.com/2012/01/newsq l-handle-big-data/>
36. P. Campaniello. The NewSQL Market Breakdown <http://www.scalebase.com/the-story-of-newsq l/>
37. Theo Haerder, Andreas Reuter. Principles of transaction-oriented database recovery. ACM Computing Surveys, Volume 15, Issue 4, December 1983, Pp. 287–317 <https://dl.acm.org/citation.cfm?id=291>
38. The Neo4j Developer Manual v3.3 <http://neo4j.com/docs/developer-manual/current/introduction/>