https://doi.org/10.31891/2219-9365-2022-70-2-1
УДК 004.65

Ольга ТАРАСЮК
Одеський технологічний університет «ШАГ»
https://orcid.org/0000-0001-5991-8631
Анатолій ГОРБЕНКО
Національний аерокосмічний університет ім. М. Є. Жуковського «ХАІ»
Університет Лідс Беккет
https://orcid.org/0000-0001-6757-1797
Андрій КАРПЕНКО
Національний аерокосмічний університет ім. М. Є. Жуковського «ХАІ»
https://orcid.org/0000-0003-2789-1168

РОЗВИТОК АРХІТЕКТУР, ТЕОРЕМ ТА МОДЕЛЕЙ ВЛАСТИВОСТЕЙ РОЗПОДІЛЕНИХ СИСТЕМ ЗБЕРІГАННЯ ІНФОРМАЦІЇ

Бази даних пройшли певний шлях еволюції від архітектури мейнфреймів до глобально-розподілених нереляційних сховищ, що призначені для збереження величезних обсягів інформації та обслуговування мільйонів користувачів. У статті зазначено драйвери та передумови цього розвитку, а також розглянуто трансформацію моделей властивостей систем керування базами даних та теорем, що формалізують відносини між ними. Зокрема, розглянуто обумовленість переходу від моделі властивостей ACID до моделі BASE, яка пом'якшує вимоги до узгодженості даних, що є необхідним для забезпечення високої продуктивності розподілених баз даних з багатьма репліками. Крім того, надано стисле обґрунтування теорем CAP та PACELC, які встановлюють взаємовиключні відносини між доступністю, узгодженістю та швидкодією у реплікованих інформаційних системах та проаналізованоо їх обмеження. Зазначено проблеми сумісності моделей узгодженості, що використовуються різними нереляційними сховищами даних, та, в якості прикладу, детально розглянуто можливі налаштування узгодженості NoSQL баз даних Cassandra, MongoDB та Azure CosmosDB. Результати еволюції архітектур розподілених баз даних узагальнено за допомогою нотації GSN (Goal Structuring Notation). Також окреслено подальші напрямки наукових досліджень та шляхи подальшого розвитку глобально-розподілених інформаційних систем та сховищ даних..

Ключові слова: розподілені бази даних, моделі ACID та BASE, теореми CAP та PACELC, база даних Cassandra, Azure CosmosDB, MongoDB, нотація GSN.

Olga TARASYUK
Odessa Technological University STEP
Anatoliy GORBENKO
National Aerospace University "Kharkiv Aviation Institute", Leeds Beckett University
Andrii KARPENKO
National Aerospace University "Kharkiv Aviation Institute"

MODELING OF THE DECISION-SUPPORTING PROCESS ON THE POSSIBILITY OF CONCLUDING THE CONTRACT ON THE THERAPEUTIC SERVICES PROVISION

Today, we live in the world of information technologies, which have penetrated into all possible spheres of human activity. Recent developments in database management systems have coincided with advances in parallel computing technologies. In view of this fact, a new class of data storage has appeared, namely globally distributed non-relational database management systems, and they are now widely used in Twitter, Facebook, Google and other modern distributed information systems to store and process huge volumes of data.

Databases have undergone a certain evolution from mainframe architecture to globally distributed non-relational repositories designed to store huge amounts of information and serve millions of users. The article indicates the drivers and prerequisites of this development, and also considers the transformation of models of properties of database management systems and theorems that formalize the relationship between them. In particular, the conditionality of the transition from the ACID property model to the BASE model is considered, which relaxes the requirements for data consistency, which is necessary to ensure the high performance of distributed databases with many replicas. In addition, a concise justification of the SAR and PACELC theorems, which establish mutually exclusive relationships between availability, consistency, and speed in replicated information systems, is provided, and their limitations are analyzed. The compatibility issues of the consistency models used by different non-relational data stores are noted, and, as an example, the possible consistency settings of the NoSQL databases Cassandra, MongoDB, and Azure CosmosDB are discussed in detail. The results of the evolution of distributed database architectures are summarized using the GSN (Goal Structuring Notation). Further directions of scientific research and ways of further developing globally distributed information systems and data repositories are also outlined.

Keywords: distributed databases, ACID and BASE models, CAP and PACELC theorems, Cassandra database, Azure CosmosBD, MongoDB, GSN notation.

Постановка проблеми у загальному вигляді
та її зв'язок із важливими науковими чи практичними завданнями

На сьогоднішній день ми живемо у світі інформаційних технологій, які проникли у всі можливі сфери життєдіяльності людства. Нещодавній розвиток систем управління базами даних співпав з досягненнями у технологіях паралельного обчислення. З огляду на цей факт з'явився новий клас сховищ даних, а саме глобально-розподілені нереляційні системи управління базами даних [1], і саме вони зараз широко застосовуються у Twitter, Facebook, Google та інших сучасних розподілених інформаційних системах для збереження та обробки величезних обсягів даних.

Системи управління базами даних пройшли певний шлях еволюції від архітектури мейнфреймів до глобально-розподілених нереляційних сховищ даних [2]. Здебільшого ця еволюція була зумовлена появою нових вимог до збереження даних та забезпечення доступу до них, пов'язаних з ростом обсягів інформації, зростанням швидкості її надходження та необхідністю одночасного обслуговування величезної кількості гетерогенних споживачів інформації.

Еволюція теорем та моделей властивостей сховищ даних. Модель ACID

Вперше акронім ACID [3] був сформований у 1983 році А. Рейтером і Т. Хардером базуючись на дослідженнях Д. Грейа. Модель ACID (Atomicity, Consistency, Isolation, Durability) представляє собою набір властивостей, які повинна мати реляційна база даних для забезпечення достовірності даних, що зберігаються:

- Atomicity (атомарність) – властивість, яка гарантує, що кожна транзакція буде виконана в повному обсязі або не виконається взагалі, не допускаючи проміжного стану;
- Consistency (узгодженість) – властивість, яка гарантує, що кожна транзакція не порушує цілісність даних у ній;
- Isolation (ізольованість) – властивість, яка гарантує, що будь-яка операція читання або запису не буде змінена іншими операціями читання або запису;
- Durability (довговічність) – властивість, яка гарантує, що зміни внесені до бази даних, які були успішно завершені, збережені назавжди, навіть у разі відмови системи, наприклад при аварійному відключенні електроживлення.

Модель BASE

BASE (Basically Available, Soft state, Eventually Consistent) [4] є більш гнучкою моделлю, ніж ACID і забезпечує ряд переваг перед ACID-сумісними базами даних, які за своїм дизайном не є у певній мірі розподіленими системами і не можуть забезпечити ефективну роботу в умовах відмов каналів зв'язку. Проте, властивості BASE не надають жорстких гарантій щодо узгодженості даних на відміну від моделі ACID:

- Basically Available (переважно доступні дані) – дані, що зберігаються, залишаються переважно доступними навіть при наявності відмов каналів зв'язку або деяких вузлів бази даних (за рахунок використання реплікації), проте узгодженість інформації не гарантується; тобто існує імовірність того, що при читанні буде повернена застаріла версія даних, або результати операції запису не будуть збережені;
- Soft state (м'який стан) – властивість, яка говорить про те, що стан системи може змінюватися навіть при відсутності нових операцій читання/запису оскільки ще можуть продовжуватися фонові процеси оновлення даних для досягнення узгодженості; таким чином, існує лише певна ймовірність того, що система (дані) знаходиться у певному стані;
- Eventually Consistent (кінцева узгодженість) – зрештою система після деякого проміжку часу досягне бажаного рівня узгодженості, та усі репліки бази даних будуть мати актуальні дані.

Модель BASE надала поштовх для розвитку нереляційних баз даних NoSQL (Non SQL, Not only SQL) [1]. Властивості, передбачені цією моделлю, забезпечують високу масштабованість сховищ даних, велику продуктивність та швидкість виконання операцій читання та запису, простоту реалізації механізмів реплікації та створюють можливість для зберігання та аналізу дійсно великих даних, так званих Big Data [5].

Теорема CAP

Гіпотеза CAP (акронім утворений з англомовних іменувань властивостей розподілених інформаційних систем: Consistency, Availability, Partition tolerance) вперше була озвучена у 2000 році Е. Брюером [6]. Згодом у 2002 році С. Гілберт та Н. Лінч довели гіпотезу та трансформували її у теорему [7].

Теорема CAP стверджує, що для будь-якої розподіленої інформаційної системи неможливо одночасно забезпечити виконання більш ніж двох із зазначених властивостей, а саме:

- Consistency (узгодженість) – властивість, яка гарантує, що кожна операція читання повертає актуальні дані, а всі вузли-репліки завжди мають однаковий набір даних.
- Availability (доступність/готовність) – властивість, яка забезпечує отримання відповіді від бази даних на кожний запит, однак актуальність повернених даних не гарантується.
- Partition tolerance (стійкість до розподілення) – здатність системи продовжувати роботу та обробляти запити навіть у разі втрати або значної затримки повідомлень між вузлами-репліками або відмови деяких реплік.

У разі відмови комунікаційних каналів і втрати зв'язку між вузлами в розподілених реплікованих системах баз даних неможливо одночасно підтримувати як доступність так і узгодженість даних. Тобто або вся операція читання повинна бути скасована для збереження узгодженості, або система буде продовжувати підтримувати доступність, а отже, нехтуючи узгодженістю.

Таким чином теорема CAP зазначає розподіл між сховищами даних, що реалізують модель ACID, та тими, що побудовані за ідеологією BASE.

Теорема PACELC

Теорема PACELC була вперше описана та сформульована Д. Абаді у 2012 році [8]. Вона є подальшим розвитком теореми CAP і стверджує, що у разі розподілення (Partition) системи виникає необхідність вибору між доступністю (Availability) та узгодженістю (Consistency), а при відсутності розподілення системи існує додатковий (Else) компроміс між часовою затримкою (Latency) та узгодженістю (Consistency). Під часовою затримкою слід вважати час, який клієнт очікує від бази даних на повернення результатів запиту на читання або підтвердження успішності виконання запису.

Таким чином теорема PACELC забезпечує більш повне відображення потенційних компромісів, пов'язаних із забезпеченням узгодженості даних у розподілених інформаційних системах.

Обмеження теорем CAP та PACELC

Теореми CAP та PACELC є, безумовно, важливими для розуміння якісного зв'язку між ключовими характеристиками розподілених систем збереження та обробки інформації. Однак, вони мають низку принципових недоліків, які обмежують коло їхнього застосування.

По-перше, зазначені характеристики за своєю природою є не бінарними, як це розглядається в теоремах CAP та PACELC, а безперервними (наприклад, час обслуговування є безперервною величиною на інтервалі (0, ∞) у той час як доступність традиційно вимірюється від 0 до 100%) або мають дискретний набір значень. Наприклад, це стосується узгодженості даних, яка може змінюватися від слабкої (weak) до строгої (strong) з багатьма можливими рівнями послабленої (relaxed) узгодженості між ними.

По-друге, вони встановлюють лише якісний зв'язок між характеристиками розподіленої системи, що обмежує практичне значення теорем і не дозволяє кількісно оцінити рівень впливу одних характеристик на інші.

По-третє, теореми CAP/PACELC передбачають знаходження компромісу між узгодженістю, доступністю та швидкодією (або ж можливістю продовження роботи у разі розподілення) лише за рахунок вибору двох з цих трьох властивостей. Однак, найбільш ефективним є знаходження більш гнучкого балансу між ними.

Крім того, зазначені теореми не відображають дуалізм узгодженості даних. З одного боку узгодженість може визначати імовірність того, що всі вузли-репліки мають однаковий набір даних, або ж надавати певні гарантії щодо узгодженості даних (наприклад строга узгодженість завжди гарантує однаковість даних, що повертаються різних користувачам). З іншого боку, рівень узгодженості може бути одним із налаштувань розподіленої інформаційної системи. Наприклад, нереляційна база даних Cassandra забезпечує узгодженість у кінцевому випадку (eventual consistency), та дозволяє встановити рівень узгодженості для кожного окремого запиту, який визначає кількість реплік (всі, одна, або більшість), яким буде спрямовано цей запит.

Нарешті, існують інші важливі характеристики які суттєво пов'язані із зазначеними у теоремах CAP та PACELC. Наприклад, це стосується відмовостійкості (fault-tolerance), довговічності (durability), енергоспоживання (energy consumption), тощо.

Еволюція архітектур сховищ даних. Клієнт-серверна архітектура

Клієнт-серверна модель – це розподілена архітектура, яка відокремлює постачальників інформаційно-обчислювальних ресурсів, тобто серверів обробки та збереження інформації, від ініціаторів запитів на обробку – клієнтів (див. рис. 1).

Рис. 1. Клієнт-серверна архітектура

Доступ до даних у клієнт-серверній архітектурі забезпечується наступним чином [9]: 1) клієнт формує та надсилає запит до сервера бази даних; 2) сервер обробляє запит і отримує необхідну інформацію з бази даних, а у разі необхідності – виконує маніпуляції з інформацією, що зберігаються у базі даних; 3) після завершення обробки запиту сервер формує відповідь та надсилає її клієнту. Таким чином, ресурси клієнтського комп'ютера не приймають участь у фізичному виконанні запиту [10].

Архітектура централізованого реплікованого кластеру

Традиційна клієнт-серверна архітектура з одним сервером має певні обмеження пов'язані з доступністю та продуктивністю. Так, у разі відмови сервера всі клієнти втрачають доступ до інформації. Крім того, існує певна межа продуктивності, після досягнення якої сервер не може впоратися з обслуговуванням нових користувачів, або ж значно зростає час обслуговування запитів. Вирішенням перелічених вище недоліків може бути реплікація даних по декількох серверах. Це дозволяє розподілити зростаюче навантаження між серверами-репліками, а також забезпечити стійкість до відмов серверів, тобто підвищити готовність системі.

Таким чином, наступним етапом розвитку розподілених систем була архітектура реплікованого клієнт-серверного кластеру [11] з репліками, розташованими у межах локальної мережі (див. рис. 2). Передумовами виникнення такої архітектури була необхідність підвищення надійності, доступності та продуктивності.

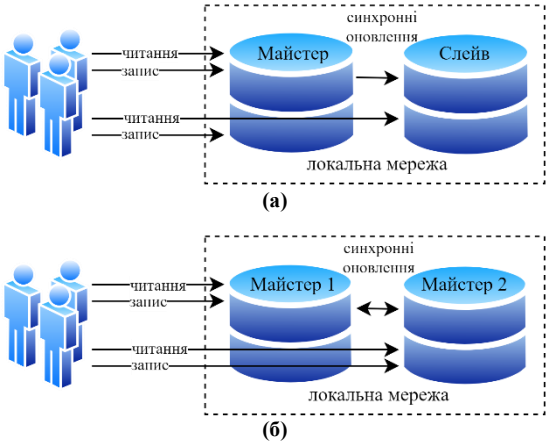

Рис. 2. Архітектура централізованого реплікованого кластеру: (а) майстер-слейв; б) майстер-майстер

Існують два різновиди реплікованого кластеру: майстер-слейв (master-slave) та майстер-майстер (master-master або multi-master). У першому випадку (рис. 2, а) всі репліки можуть обробляти запити на читання інформації, проте будь-яка операція, яка вимагає оновлення даних, повинна проходити та оброблятися майстер-сервером, який розповсюджує оновлення на всіх підлеглих серверах.

Архітектура майстер-слейв досить проста у реалізації, але до її недоліків можна віднести наявність єдиної точки відмови (майстер-сервера) та проблеми продуктивності при зростанні кількості операцій запису [12]. З цієї причини реплікація майстер-слейв переважно використовується в інтенсивних для читання додатках. Якщо майстер виходить з ладу, один із підлеглих серверів може прийняти на себе роль майстра. Однак на час вибору нового майстра кластер не може обслуговувати запити на оновлення інформації.

Наступним етапом розвитку баз даних стала поява архітектури кластеру майстер-майстер (рис. 2, б), яка є більш складною у реалізації, але вирішує недоліки попередньої схеми. Однак можливість оновлення бази даних декількома серверами одночасно викликає складність у підтримки узгодженості даних. Конфлікт може виникнути, якщо кілька серверів намагаються одночасно оновити один і той самий об'єкт у базі даних. Для виявлення та вирішення таких конфліктів використовуються спеціальні алгоритми та протоколи [13].

Треба також зазначити, що у централізованому кластері всі репліки компактно розміщені у локальній мережі, а синхронізація між майстром та іншими репліками є синхронною. Тобто, операція запису чи оновлення даних вважається успішною тільки після завершення оновлення інформації на всіх серверах-репліках. Також, всі репліки призупиняють обслуговування запитів до закінчення синхронізації даних між ними використовуючи механізм двофазних транзакцій (two-phase commit protocol [14]). Однак, оскільки всі репліки об'єднані локальною мережею, синхронізація виконується дуже швидко, і блокування роботи системи не займає багато часу.

Архітектура глобально-розподіленого реплікованого кластеру

Архітектура централізованого реплікованого кластеру ефективно обслуговує клієнтів, що розташовані поруч із серверами. Однак, при обслуговуванні географічно віддалених клієнтів виникають суттєві затримки, пов'язані зі зростанням часу пересування результатів обробки через глобальну мережу Інтернет. Також централізований кластер не захищений від відмов із загальними причинами, тобто не забезпечує катастрофостійкості [15].

Таким чином, для обслуговування географічно-віддалених користувачів виникла необхідність у побудові глобально-розподілених інформаційних систем за архітектурою майстер-майстер в яких репліки розташовані близько до груп користувачів, які вони обслуговують, а не одна до одної (рис. 3).

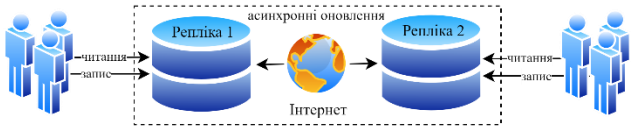

Рис. 3. Архітектура глобально-розподіленого реплікованого кластеру

Однак, така архітектура фактично унеможливлює виконання синхронних оновлень між репліками, оскільки реалізація протоколу двофазних транзакцій між глобально-розподіленими репліками призводить до катастрофічного зниження продуктивності усієї системи.

Отже, у таких системах використовується механізм асинхронних реплікацій, у якому зміни інформації в одній репліці розповсюджуються на інші репліки у фоновому режимі через деякий час. Таким чином, асинхронна реплікація призводить до появи тимчасової неузгодженості даних між вузлами-репліками.

Фактично, це обумовлює перехід від моделі ACID зі строгою узгодженістю даних, до моделі BASE, яка забезпечує тільки кінцеву узгодженість (Eventual Consistency). Як наслідок, деякі клієнти можуть отримати застарілі дані.

Безумовно, у деяких додатках послаблення вимог до узгодженості може спричинити катастрофічні наслідки, що є неприйнятним. Однак для систем, які можуть собі дозволити послаблення узгодженості, наприклад, у соціальних мережах, впровадження асинхронної реплікації дозволяє досягти високої продуктивності та зробити систему високо-масштабованою [13].

Еволюція моделей узгодженості даних

Моделі узгодженості даних визначають гарантії, що надаються сховищем даних користувачам щодо результатів операцій читання та запису при наявності конкуруючих запитів від багатьох користувачів.

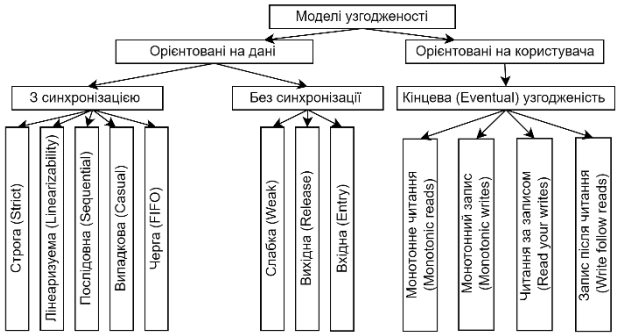

Рис. 4. Моделі узгодженості даних

Існує багато різних моделей узгодженості, які використовуються у розподілених обчислювальних системах і системах зберігання даних, які узагальнені на рис. 4. Умовно їх можна поділити на дві основні групи [16]. Перша група забезпечує певний стан сховища даних для всіх користувачів (моделі орієнтовані на дані); друга надає гарантії лише для окремого користувача (моделі орієнтовані на користувача) у той час як інформація, що повертається у відповідь на той же самий запит читання, отриманий одночасно від декількох користувачів, може відрізнятися від користувача до користувача.

Строга узгодженість не може бути ефективно досягнута у реплікованих глобально-розподілених системах. Таким чином, багато сучасних нереляційних сховищ даних реалізують різні види моделей ослабленої узгодженості.

На жаль, на сьогодні не існує загальних правил для послаблення узгодженості і, виходячи з цього, різні постачальники баз даних реалізують різні моделі узгодженості, які несумісні між собою та які важко порівняти.

Наприклад, Apache Cassandra, яка підтримує архітектуру майстер-майстер, визначає дискретний набір рівнів узгодженості, вказуючи кількість реплік, що викликаються для виконання кожній операції читання та запису (див. табл. 1). Строга узгодженість, яка гарантує, що читання завжди відображатиме найбільш актуальні дані, досягається коли сума вузлів, що викликаються при операціях запису та читання є більшою за коефіцієнт реплікації даних [17].

Модель узгодженості MongoDB, що є системою майстер-слейв, заснована на налаштуванні параметрів w та j, як показано у табл. 2. Оновлення даних з первинної репліки (майстра) на вторинні (слейви) завжди виконуються асинхронно. За замовчуванням читання виконуються з майстра, але ж читання з вторинних реплік автоматично робить MongoDB системою з кінцевою (eventual) узгодженістю [17].

Azure CosmosDB – це хмарна система зберігання даних, репліки якої можна розмістити по всьому світу у кількох регіонах Azure. CosmosDB підтримує п’ять рівнів узгодженості, характеристика яких наведена у табл. 3, та може бути налаштована для роботи з однією або кількома майстер-репліками [17].

Таблиця 1

Модель узгодженості Apache Cassandra

Рівень узгодженості	Визначення та гарантії узгодженості
ONE	Дані мають бути записані до журналу транзакцій та у пам’ять принаймні однієї репліки перш ніж підтвердити успішність операції запису клієнту; під час зчитування даних Cassandra запитує та повертає відповідь з однієї репліки (використовується найближча репліка з найменшою затримкою); строга узгодженість забезпечується, коли сума реплік, що викликаються для читання та запису даних більша за коефіцієнт реплікації
TWO	Дані повинні бути записані принаймні на дві репліки; операція читання повертає найбільш актуальний запис із двох найближчих реплік (найбільш актуальні дані визначаються шляхом порівняння часових міток записів, повернутих цими двома репліками)
THREE	Подібно до TWO але для трьох реплік
QUORUM	Кворум (тобто більшість) вузлів-реплік повинен підтвердити запис або повернути відповідь на запит читання
ALL	Дані повинні бути записані на всі вузли репліки в кластері перед підтвердженням успішності операції; запит на читання завжди повертає найбільш актуальні дані після того, як отримання відповіді від усіх реплік; операція читання/запису не буде вважітися успішною, якщо навіть одна репліка не відповідсть на запит
EACH_QUORUM, LOCAL_QUORUM LOCAL_ONE	Додаткові рівні узгодженості, які використовуються, якщо репліки кластера Cassandra розподілені по декількох центрах обробки даних

Таблиця 2

Модель узгодженості MongoDB

Рівень узгодженості	Визначення та гарантії узгодженості
w:0, j:false	Найслабший рівень узгодженості (записи можуть бути втрачені навіть без розподілу системи), що забезпечує найменшу затримку читання/запису
w:1, j:false	Гарантовано запис на диск первинної репліки; це забезпечує досить низьку затримку, але й дуже слабку узгодженість
w:2, j:false	Гарантовано запис на диск первинної репліки та в пам'ять однієї з вторинних реплік; це забезпечує низьку затримку та слабку узгодженість
w:2, j:true	Гарантовано запис на дисках первинної репліки та однієї з вторинних реплік; це забезпечує середню затримку та узгодженість
w:majority, j:false	Гарантовано запис на дисках первинної репліки та в пам’ять більшості вторинних реплік; це забезпечує середню затримку та узгодженість
w:majority, j:true	Гарантовано запис на дисках первинної репліки та більшості вторинних реплік; це забезпечує високу узгодженість даних, але ж тягне й високі часові затримки на виконання операцій читання та запису

Таблиця 3

Модель узгодженості Azure CosmosDB

Рівень узгодженості	Визначення та гарантії узгодженості
STRONG (Reads: local minority; Writes: global majority)	Строга узгодженість забезпечує гарантії лінеаризованості, тобто, наприклад, читання гарантовано повертає найбільш актуальні результати запису/оновлення даних
BOUNDED STALENESS (Reads: local minority: Writes: local majority)	Гарантується, що читання завжди повертатиме впорядковані результати оновлення даних. Однак, допускається, що читання може відставати від запису, але не більше ніж на *K* оновлень або на *T* часовий інтервал (тобто вікно нестачі (staleness window)), залежно від того, що менше.
SESSION (Reads: single replica with session token; Writes: local majority)	Гарантується, що читання завжди повертатиме впорядковані результати оновлення даних протягом однієї клієнтської сесії; для інших клієнтів також гарантоване монотонне читання та запис, запис після читання (write-follows-reads) та читання свого запису (read-your-writes)
CONSISTENT PREFIX (Reads: single replica; Writes: local majority)	Гарантується, що читання завжди повертатиме впорядковані результати оновлення даних без пробілів; наприклад, гарантується дотримання впорядкованої послідовності оновлень даних (починаючи з першої), яка зберігалася в мастер-репліці в якийсь час у минулому; натомість, допускається, що найбільш актуальні дані можуть бути відсутніми
EVENTUAL (Reads: Single replica; Writes: local majority)	Немає гарантії порядку читання; за відсутності подальшого запису репліки зрештою сходяться; користувачі можуть читати дані, які «старші», ніж ті, що були прочитані раніше

Еволюція моделей узгодженості даних

GSN (Goal Structuring Notation) – схема, яка ілюструє окремі аргументи певного судження та його співвідношення, які існують між цими елементами. Основні елементи схеми є:

- Мета (Goal, G1);
- Рішення (Solution, Sn1);
- Стратегія (Strategy, Str1);
- Контекст (Context, C1);
- Припущення (Assumption, A1);
- Доказ (Justification, J1).

Формат GSN закріплений у стандарті «Goal Structuring Notation (GSN) Community Standard» [18]. Об'єднання елементів у єдину структуру описується як «goal structure». Ця структура показую, як цілі послідовно декомпозуються на підцілі, перш ніж буде досяжна точка, у якої вони можуть бути підтверджені прямим посиланням на рішення. У рамках цієї декомпозиції за допомогою GSN також можна пояснити прийняті стратегії аргументації, обґрунтування підходу та контекст, у якому сформульовані цілі. Узагальнений процес еволюції архітектур сховищ даних, формалізований за допомогою нотації GSN, представлений на рис. 5.

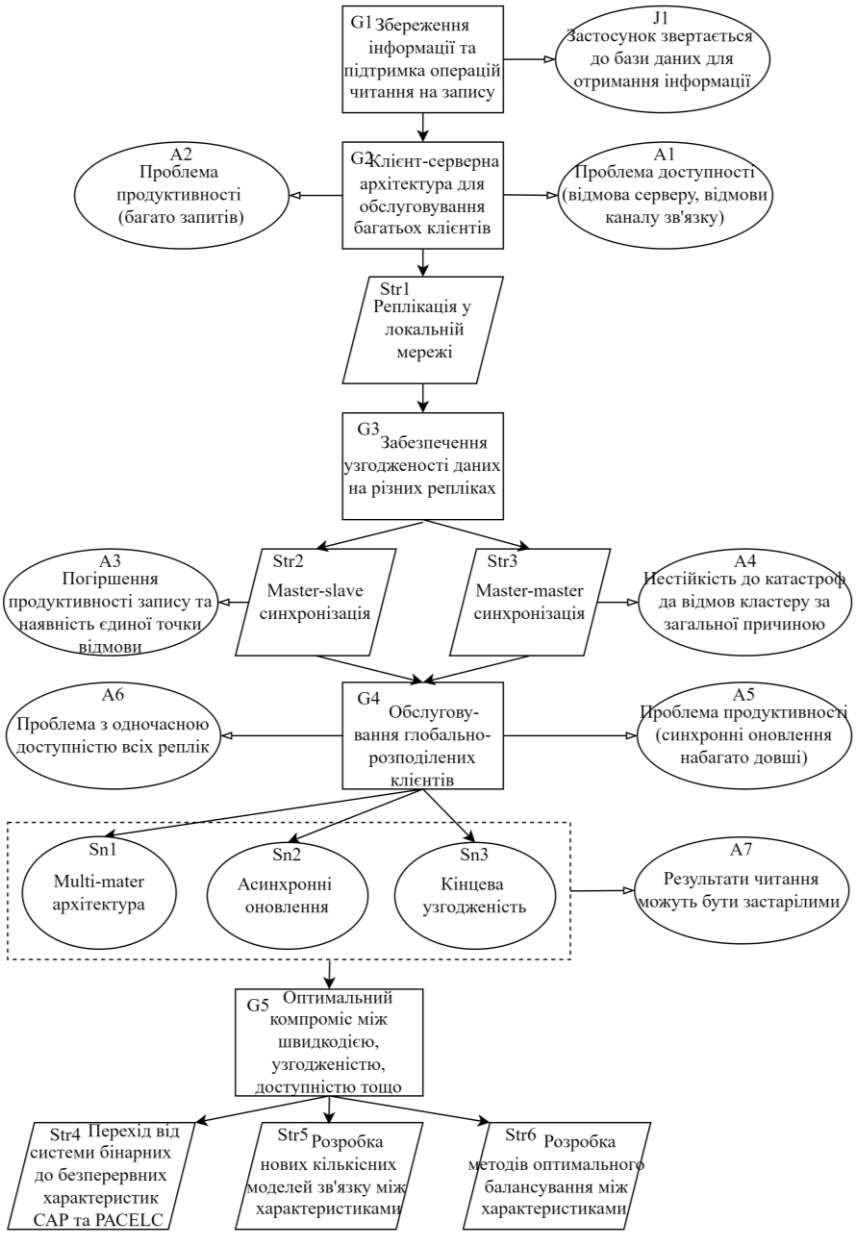

Рис. 5. Еволюція архітектур сховищ даних у нотації GSN

Збереження інформації та підтримка маніпуляцій з ними є основною метою (G1) створення сховищ даних. Мета G2 уособлює необхідність обслуговування багатьох користувачів з конкуруючими запитами. Мета G2 тягне необхідність вирішення проблем надійності/доступності (A1) та продуктивності (A2). Ці проблеми вирішуються за допомогою введення допоміжних серверів-реплік, що зберігають копію даних та (Str1). Це забезпечує відмовостійкість та дозволяє підвищити продуктивність сховища даних шляхом розподілу клієнтських запитів між усьома репліками. Наступна мета пов'язана з необхідністю синхронізації даних між репліками (G3). Для її досягнення було запроваджено дві архітектури баз даних: master-slave (Str2) та master-master (Str3). Реалізація стратегії Str2 є набагато простішою, але залишає проблему з недостатньою продуктивністю операцій запису та наявністю єдиної точки відмови (A3). Стратегія Str3 більш складна в реалізації і не вирішує проблему катастрофостійкості та можливості відмови всього

кластера через загальну причину (A4), оскільки всі репліки розташовані в локальній мережі близько один до одного, що необхідно для швидкої синхронізації даних між ними. Додаючи наступну ціль (G4), яка спрямована на підтримку обслуговування глобально-розподілених користувачів завдяки розміщенню реплік у різних географічних локаціях, з'являються додаткові проблеми: проблема повільної синхронізації даних через мережу Інтернет між віддаленими репліками (A5), проблема частого розподілу кластера у разі відмови або недоступності через нестабільність мережевого середовища Інтернет хоча б однієї з реплік (A6).

Для їх вирішення було запропоновано такі підходи: використання архітектури з кількома майстрами (multi-master) (Sn1), впровадження асинхронних оновлень між репліками (Sn2) та гарантію тільки кінцевої узгодженості даних (Sn3). Ці фундаментальні рішення лягли в основу створення та швидкого розвитку сучасних нереляційних баз даних NoSQL. Однак ці рішення створили передумови для появи іншої проблеми, яка й досі залишається невирішеною (A7), а саме – принципову можливість отримання користувачами застарілих/неактуальних результатів при читанні інформації зі сховища даних.

Таким чином виникає необхідність досягнення нової мети (G5), яка полягає у знаходженні оптимального балансу між часовими затримками, узгодженістю даних, доступністю та іншими характеристиками розподілених баз даних.

Для її досягнення, по-перше, пропонується розглядати характеристики розподілених реплікованих сховищ даних, які використовуються у теоремах CAP та PACELC (зокрема, часові затримки, доступність та узгодженість), як безперервні, а не дискретні величини (Str4). По-друге, необхідно розробити нові аналітичні моделі, які б дозволили встановити кількісний зв'язок між зазначеними характеристиками Str5 (на відміну від теорем CAP та PACELC, які встановлюють лише якісне співвідношення між ними та декларують можливість вибору лише двох з цих трьох характеристик при створені розподіленої інформаційної системи). По-третє, актуальним є розробка нових методів знаходження оптимального або ж раціонального компромісу між характеристиками розподілених баз даних, які є суперечливими за своєю природою (Str6).

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямі

1) Еволюція баз даних тісно пов’язана з досягненнями інформаційних технологій та розвитком комп’ютерних систем. З ростом числа оброблюємої інформації виникали й нові моделі та архітектурні рішення, які використовувалися для створення сховищ даних. Наприклад, при побудові глобально-розподілених систем зберігання великих даних (Big Data) модель ACID, яка домінувала протягом десятиліть, була замінена моделлю BASE, а від гарантій строгої узгодженості (strong consistency) був здійснений перехід до понять ослабленої (relaxed) або кінцевої (eventual) узгодженості.

2) Неможливість одночасного забезпечення стійкості до розподілення, готовності та строгої узгодженості даних у глобально-розподілених реплікованих інформаційних системах вперше була зазначена у теоремі CAP у 1999 році. З годом, ця теорема набула подальшого розвитку у вигляді теореми PACELC, яка визначає необхідність додаткового вибору між швидкодією та узгодженістю під час проектування та експлуатації таких систем.

3) У статті розглянуто передумови розвитку теорем та моделей властивостей сховищ даних, етапи еволюції архітектур розподілених інформаційних систем, а також різні моделі узгодженості даних, що використовуються сучасними нереляційними базами даних NoSQL. Крім того, зазначені напрямки та актуальні завдання наукового та прикладного характеру для подальшого розвитку глобально-розподілених інформаційних систем.

References

1. NoSQL: past, present, future [Електронний ресурс]. – QCon – Режим доступу – https://www.infoq.com/presentations/NoSQL-History/

2. Meier, A. SQL and NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management [Text] / A. Meier, M. Kaufmann. – Berlin: Springer Verlag, 2019. – 229 p.

3. Medjahed B., Generalization of ACID Properties [Текст] / B. Medjahed, M. Ouzzani, A.K. Elmagarmid // Encyclopedia of Database Systems. – 2009. – pp. 1221-1222.

4. Pritchett, D. BASE: An Acid Alternative [Текст] / D. Pritchett // ACM Queue. – 2008. – Vol. 6, No. 3. – P. 48-55.

5. Ugur N.G., Understanding Big Data[Текст] / N.G. Ugur, A.H. Turan // Advances in Data Mining and Database Management. – 2020. – pp. 1-29.

6. Brewer E., Towards robust distributed systems [Текст] / E. Brewer // 19th Annual ACM Symposium on Principles of Distributed Computing. – Portland, USA, 2000. – P. 7-8.

7. Gilbert, S., Brewer’s Conjecture and the Feasi- bility of Consistent, Available, Partition-Tolerant Web Services [Текст] / S. Gilbert, N. Lynch // ACM SIGACT News. – 2002. – Vol. 33, No. 2. – P. 51-59.

8. Abadi, D., Consistency tradeoff in modern distributed database system design [Текст] / D. Abadi // IEEE Computer. – 2012. – Vol. 45, No. 2. – P. 37-42

9. Distributed Database Systems [Електронний ресурс]. – University of Cape Town – Режим доступу: https://www.cs.uct.ac.za/mit_notes/database/pdfs/chp15.pdf

10. Understanding Distributed Systems [Електронний ресурс]. – Oracle docs – Режим доступу: https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SD173/ch1.htm

ISSN 2219-9365

11. Client-Server Model [Електронний ресурс]. – GeeksforGeeks – Режим доступу: https://www.geeksforgeeks.org/client-server-model/

12. Strickland, R. Cassandra High Availability [Текст] / R. Strickland. – Packt, 2014. – 186 p.

13. Multi-master Replication Solutions for PostgreSQL [Електронний ресурс]. – Percona – Режим доступу: https://www.percona.com/blog/2020/06/09/multi-master-replication-solutions-for-postgresql/

14. Al-Houmaily J. Y., Two-Phase Commit [Текст] / Y. J. Al-Houmaily, G. Samaras // Encyclopedia of Database Systems. – 2009. – pp. 3204-3209

15. Choy M., Disaster recovery techniaues for databasesystems [Текст] / M. Choy, H.V. Leong, M. H. Wong // Communications of the ACM. – 2000. – Vol.43. – P. 6.

16. Database replication 101 Systems [Електронний ресурс]. – OVHcloud – Режим доступу: https://blog.ovhcloud.com/database-replication-101/

17. NoSql Database Architectural Comparison [Електронний ресурс]. – Fixstars – Режим доступу: https://griddb.net/en/docs/NoSQL_Database_Architectural_Comparison.pdf

18. Goal Structuring Notation [Електронний ресурс]. – Safety-Critical Systems Club – Режим доступу: https://scsc.uk/gsn