УДК 004.94 http://doi.org/10.15407/pp2022.02.057

В.А. Резніченко

60 РОКІВ БАЗАМ ДАНИХ (четверта частина)

Наводиться огляд досліджень і розробок баз даних із моменту їх виникнення в 60-х роках минулого століття і до сьогодні. Виділяються наступні етапи: виникнення і становлення, бурхливий розвиток, епоха реляційних баз даних, розширені реляційні бази даних, постреляційні бази даних і великі дані. На етапі становлення описуються системи IDS, IMS, Total і Adabas. На етапі бурхливого розвитку висвітлені питання архітектури баз даних ANSI/X3/SPARC, пропозицій КОДАСИЛ, концепції і мов концептуального моделювання. На етапі епохи реляційних баз даних розкриваються результати наукової діяльності Е. Кодда, теорія залежностей і нормальних форм, мови запитів, експериментальні дослідження і розробки, оптимізація та стандар¬тизація, управління транзакціями. Етап розширених реляційних баз даних присвячений опису темпоральних, просторових, дедуктивних, активних, об'єктних, розподілених та статистичних баз даних, баз даних масивів, машин баз даних і сховищ даних. На наступному етапі розкрита проблематика постреляційних баз даних, а саме NOSQL-, NewSQL- і онтологічних баз даних. Шостий етап присвячений розкриттю причин виникнення, характерних властивостей, класифікації, принципів роботи, методів і технологій великих даних. Нарешті, в останньому розділі подано короткий огляд досліджень і розробок із баз даних у Радянському Союзі.

Ключові слова. Типи баз даних: ієрархічна, мережева, реляційна, навігаційна, темпоральна, просторова, просторово-темпоральна, просторово-мережева, об'єктів, що переміщуються, дедуктивна, активна, об'єктно-орієнтована, об'єктно-реляційна, розподілена, паралельна, масивів, статистична, багатовимірна, машина баз даних, сховища даних, NoSQL, ключ-значення, стовпчикова, документно-орієнтована, графова, мультимодельна, хмарна, наукова, багатозначна, XML, NewSQL, онтологічна, великі дані.

Етап 5. Постреляційні бази даних (2000 – 2010+)

Проникнення Інтернету в усі сфери нашого життя викликало суттєве зростання кількості джерел даних, неймовірне збільшення їх обсягу та інтенсивності використання. Це призвело до проблем зберігання, обробки і до труднощів з оперуванням неструктурованою інформацією. Класичні реляційні СУБД, які вірно служили людству протягом 40 років, виявилися неспроможними впоратися із таким новим викликом. У зв'язку з цим з'явився новий напрямок у БД, який дістав назву NoSQL, в результаті з'явилися БД ключ – значення, документні, колончаті, графові. Однак прихильники реляційних БД вирішили не здаватися без бою і спрямували свої зусилля на таку модернізацію реляційної концепції, за якої вдалося б подолати цей виклик початку нового століття. Так виник напрямок NewSQL. Тоді ж виникла концепція семантичного вебу, мета якого – підвищити семантику вебу задля створення механізмів більш релевантного пошуку. Найважливішою складовою такого вебу стало поняття онтології. Одним із варіантів збереження онтологій стали онтологічні бази даних. Зрештою в цей період почали бурхливо розвиватися повнотекстові БД та електронні бібліотеки. Стислому аналізу цих складових постреляційних БД і присвячено даний розділ.

NoSQL – бази даних

У 2000-і роки з появою вебресурсів із величезними сховищами різнорідної інформації дослідники почали частіше аналізувати нові структури даних. Реляційний підхід заснований на чіткому структуруванні даних і строго формалізованому доступі до них. Це робить БД негнучкими і тим самим уповільнює швидкість роботи. Новий підхід базувався на відмові від фіксованої схеми даних і мови SQL.

NoSQL – термін, що означає низку підходів, спрямованих на реалізацію СУБД, які мають суттєві відмінності від

моделей, котрі використовуються в традиційних реляційних СУБД із доступом до даних засобами мови SQL.

Історія терміну NoSQL. Термін NoSQL уперше з'явився 1998 року й використовувався для опису реляційної БД, розробленої Карлом Строцці (Carlo Strozzi) [755]. Вона не використовувала SQL як мову запитів. Це початкове використання даного терміну нічого спільного не має із сучасною технологією NoSQL. Водночас у першому десятиріччі 21-го століття з'явилися Neo4j (2000), Google BigTable (2004), CouchDB (2005), Amazon Dynamo (2007), Hypertable (2007), Hbase (2007), Dynomite (2008), Voldemort (2009), Cassandra (2009), MongoDB (2009), що були нереляційними СУБД.

Карло Строцци

2009 року в Сан-Франциско Йохан Оскарссон (Johan Oskarsson) організував семінар для обговорення нових технологій із збереження й обробки даних [756]. Головним стимулом зустрічі була поява на ринку розподілених нереляційних продуктів. Ерік Еванс (Eric Evans) запропонував зробити яскравою вивіскою семінару ємний і лаконічний термін "NoSQL" [757]. Термін планувався лише на одну зустріч і не мав якогось глибокого смислового навантаження. Але сталося так, що він розповсюдився світовою мережею і став де-факто назвою цілого напряму в IT-індустрії. Водночас термін NoSQL не означає якусь одну конкретну технологію чи продукт. Наімовірніше він характеризує вектор розвитку IT у бік від реляційних баз даних.

Ерік Еванс

Властивості NoSQL баз даних.

Існує багато різних типів NoSQL БД, та для більшості з них характерні такі властивості:

- **Гнучкість.** Зазвичай бази даних NoSQL пропонують гнучкі схеми, що дозволяє здійснювати розробку швидше й уможливлює поетапну реалізацію. Завдяки використанню гнучких моделей даних БД NoSQL добре підходить для частково структурованих і неструктурованих даних.

- **Горизонтальна (еластична) масштабованість.** Бази даних NoSQL розраховані на масштабування з використанням розподілених кластерів апаратного забезпечення, а не за рахунок додавання дорогих надійних серверів. Висока доступність за рахунок слабкої узгодженості (за рахунок спрощеної семантики ACID).

- **Висока продуктивність.** Бази даних NoSQL оптимізовані для конкретних моделей даних і механізмів доступу, що дозволяє досягнути вищої продуктивності порівняно із реляційними базами даних.

- **Широкі функціональні можливості.** БД NoSQL надають API та типи даних із широкою функціональністю, які спеціально розроблені для відповідних моделей даних.

- **Слабоструктуровані (schemaless) дані.** Структура даних не регламентована. Її можна динамічно змінювати.

- **Підтримка агрегатів.** NoSQL сховища оперують не лише атомарними, а й агрегатними об'єктами. В такому разі не потрібні нормалізовані відношення.

- **Розподіленй системи,** зазвичай без централізованого управління (децентралізовані).

- **Підтримка розподілених систем** без спільно використовуваних ресурсів (share nothing).

Аби зрозуміти, чого можна досягти внаслідок розробки і використання баз даних, було сформульовано твердження Брюера.

Теорема CAP, відома також як теорема Брюера (Brewer), - евристичне ствердження того, що в будь-якій реалізації розподілених обчислень можливо забезпечити не більше двох із трьох наступних властивостей:

- **узгодженість даних (consistency) -** в усіх обчислювальних вузлах у певний момент часу дані не суперечать одне одному (семантика ACID);

- **доступність (availability)** – будь – який запит до певної системи завершується коректною відповіддю, однак без гарантії, що відповіді всіх вузлів системи співпадають;

- **стійкість до розділення (partition tolerance)** – поділ розподіленої системи на кілька ізольованих секцій не призводить до некоректності відгуку від кожної із секцій.

Принцип був запропонований професором Каліфорнійського університету в Берклі Еріком Брюером (Eric Brewer) і Армандо Фоксом (Armando Fox) 1999 року [758], а 2002 року це твердження довели Сет Гільберт (Seth Gilbert) і Ненсі Лінч (Nancy Lynch) [759].

Ерік Брюер

Згодом теорема здобула широку популярність і визнання серед спеціалістів із розподілених обчислень. Концепція NoSQL, у межах якої створюються розподілені системи управління базами даних, часто-густо використовує цей принцип як обгрунтування неминучості відмови чи-то від узгодженості даних, чи від доступності.

На рис. нижче наводиться графічне представлення, на якому сторони трикутника відповідають парам властивостей теореми CAP, поруч наводяться приклади систем, що відповідають цим властивостям.

Класифікація систем згідно CAP

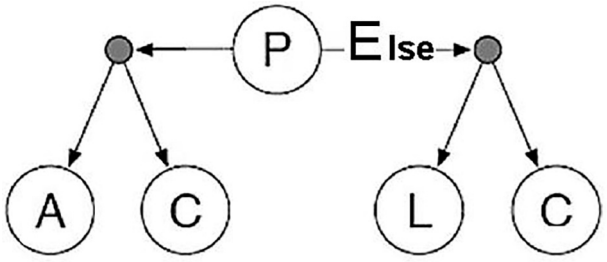
Теорема PACELC. Ця теорема, як і CAP, описує, які саме обмеження й компромісні рішення мають розподілені системи щодо узгодженості, доступності й стійкості до розподілення.

Проте, теорема PACELC додатково стверджує, що необхідно йти на компроміс між затримками отримання відповіді та узгодженням навіть за відсутності стійкості до розділення, що забезпечує краще уявлення про можливі компроміси для розподілених систем.

У згаданій теоремі використовуються не просто трикутник CAP, а наступний умовний вислів:

якщо (**P**)artition tolerance
то(**A**)valability або(**C**)onsistency
інакше (**L**)atency чи (**C**)onsistensy

Простіше кажучи, за умови стійкості до розділення (P) можна обрати (A) або узгодженість (C) (це теорема CAP), інакше (E)lse, якщо стійкості до розділення немає, можна вибрати між часом затримки (L) або узгодженістю ©. Ця ситуація графічно представлена на рис. нижче.



У такий спосіб ця теорема дає чотири різновиди розподілених систем:

PA/EL – висока доступність (A) за умови стійкості до розподілення (P) інакше (E), висока швидкість відповіді (L) (Dynamo, Cassandra, Cosmos DB, Riak);

PC/EC – в обох випадках обирається висока узгодженість (C) (Couchbase, VoltDB/H-Store, Megastore);

PC/EL – висока узгодженість (C) за умови стійкості до розподілення (P) інакше (E) висока швидкість відповіді (L) (PNUTS);

PA/EC – висока доступність (A) за умови стійкості до розподілення (P) інакше (E) висока узгодженість (C) (MongoDB).

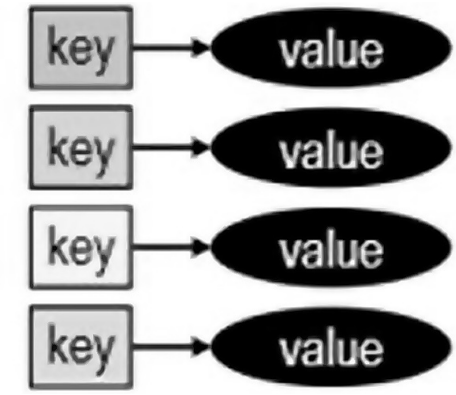
Теорема PACELC уперше була представлена в Інтернеті Даніелем Дж. Абаді (Daniel J. Abadi) з Йельського університету 2010 року, а згодом опублікована у вигляді статті 2012 року [760]. 2015 року він був нагороджений премією Very Large Data Base Endowment Inc. (VLDB) за найкращу статтю за попередні 10 років. 2020 року Даніель Дж. Абаді отримав звання «Дійсний член ACM» (ACM Fellow) за «великий внесок у потокові, розподілені, графові й колончаті бази даних».

Даніель Дж. Абаді

Типи NoSQL баз даних. Далі наводяться різні типи NoSQL баз даних, вказується на кілька з них, що належать кожному з типів. У [761] наводиться велика кількість різних систем класифікації NoSQL із широким зазначенням NoSQL баз даних, приналежних кожному з типів.

Модель асоціативного масиву (associative array model). Являє собою пару «ключ – значення». Асоціативний масив відображає ключ на значення, тобто асоціює значення з ключем. Значенням може бути як атомарна одиниця даних, так і складніша конструкція, як-от список. Формальний опис цієї моделі й алгебри наведено в [762, 763]. Праця [764] присвячена дослідженню застосування цієї моделі в базах даних.

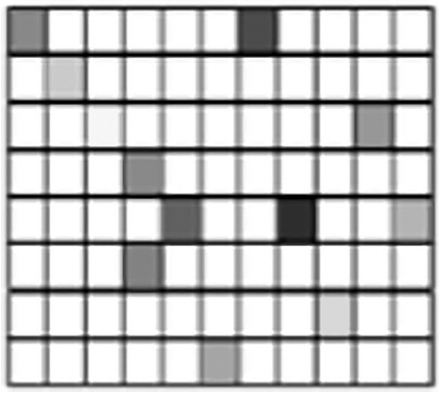
БД ключ – значення (key – value data base). Асоціативні масиви лягли в основу так званих баз даних «ключ – значення». Вважається, що ключ має бути унікальним.



Згідно із [765] першою системою баз даних типу «ключ – значення» була створена 1986 року система GT.M (Greystone Technology M) [766], призначена для високопродуктивної обробки транзакцій. За час, що відтоді минув, було створено безліч систем баз даних цього типу. На сайті DB – engines наведено 57 систем баз даних типу «ключ – значення» [767] і станом на січень 2022 року до трійки найпопулярніших належали Redis, Amazon Dynamo DB, Microsoft Azure Cosmos DB. Додаткова інформація щодо використання баз даних «ключ – значення» міститься в розділі «Великі дані».

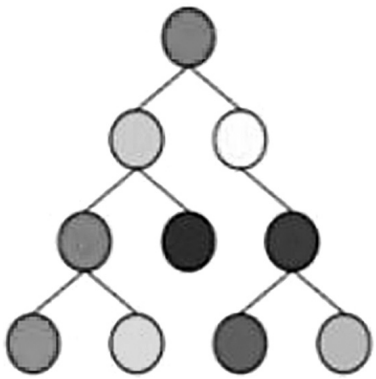
Модель триплетів (triple store model). Розширенням асоціативного масиву стала модель триплетів – три елементи, з'єднані у вираз «суб'єкт – предикат - об'єкт» [768], або в класичний елемент інформації «об'єкт – атрибут – значення». По суті триплет – це одна комірка класичного відношення. Завдяки зазначенню властивості, для якої наведено значення, триплети є інформативнішими за пари «ключ – значення». Кожному об'єкту може відповідати стільки триплетів, скільки властивостей має даний об'єкт. За адресою https://en.wikipedia.org/wiki/Comparison_of_triplestores наведено широкий список баз даних різних сховищ триплетів.

Колончаті БД (column – oriented databases, wide column store / column families). Бази даних, засновані на триплетах, дістали назву колончатих (поколончатих або баз даних, що складаються із сімейства стовпців).



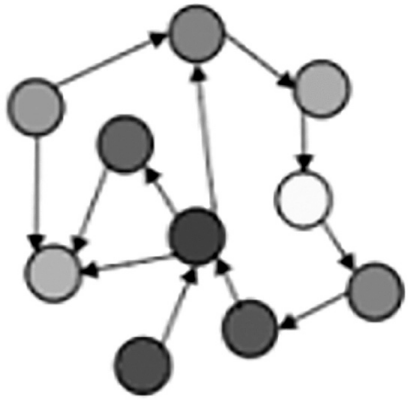
Назва пояснюється тим, що, зібравши разом усі триплети з однаковою властивістю (атрибутом), отримуємо одну колонку відношення. Представниками колончатих СУБД є DynamoDB, Google BigTable Cassandra, Scylla, HBase, Hypertable Mulgara, PNUTS тощо. Детальніше розкриття NoSQL баз даних цього типу наводиться в розділі «Колончаті бази даних».

Документно – орієнтовані БД (document – oriented databases). Об'єднання триплетів, що описують один об'єкт, називається документом. Значеннями можуть бути рядки, числа, масиви та інші вкладені триплети. Значення можуть вкладатися багаторазово. Бази даних, засновані на документах, дістали назву документно – орієнтованих. Зразками СУБД цього типу є IBM Domino, RavenDB, CouchDB, ThruDB, MongoDB, DocumentDB тощо. Детальніше про NoSQL баз даних цього типу можна ознайомитися в розділі «Документно – орієнтовані бази даних».



Графові БД (graph databases). Триплету також надається семантика «об'єкт – відношення - об'єкт». Такий триплет призначений для зберігання інформації,

яка в традиційних базах даних називається зв′язком. Представлення зв′язків між конкретними об′єктами дає змогу описати предметну область у вигляді семантичної мережі або графа, в якому об′єкти утворюють вузли, а відношення – дуги або ребра.



Такі БД дістали назву графових (graph database). Варто підкреслити, що графові моделі не новина: опис схожих баз даних можна знайти в монографії [2], виданій ще 1985 року. Там вони згадуються як бінарні бази даних. Завдяки інтуїтивній простоті й придатності для опису слабо структурованої інформації графові СУБД (Neo4j, AllegroGraph, InfiniteGraph, HyperGraphDB, OrientDB та інші) активно завойовують ринок. Детальніше розкриття NoSQL баз даних цього типу наводиться в розділі «Графові бази даних».

Структура GLOBAL

Зазначимо, що об′єкти, асоціативні масиви, триплети й документи – поняття схожі. В [769] розглядається універсальна структура GLOBAL, за допомогою якої можна представити будь – який із цих елементів. GLOBAL використовується у відомій СУБД Cashe.

Згідно з інформацією nosql-database.org, на січень 2022 року в світі існує понад 225 систем управління NoSQL – базами даних. На сайті наведено список цих СУБД із доволі повною їх класифікацією. Ця класифікація, окрім наведених вище типів NoSQL – баз даних, містить також:

- мультимодельні БД;
- об′єктні БД;
- БД мереж;
- хмарні БД;
- БД XML;
- багатовимірні БД;
- багатозначні БД;
- БД джерел подій (event sourcing);
- БД часових послідовностей / потокові БД (time series/streaming);
- Наукові й спеціалізовані БД.

Мови запитів. Була запропонована велика кількість NoSQL мов запитів, як-от AQL - ArangoDB Query Language, CQL - Cassandra Query Language, HQL - Hypertable Query Language, Cypher – мова запитів графічної бази даних Neo4j. Окрім того, компанія Couchbase, що розвиває такі системи, як Couch DB, Memcached і Membase, анонсувала створення нової мови запитів – UnQL (Unstructured Data Query Language) [770]. Вона являє собою надмножину класичної SQL, тобто багато в чому з нею сумісна, однак орієнтована на роботу з неструктурованими даними. Проєкт виконано спільними зусиллями Річарда Гіппа (Richard Hipp), творця SQLite, та Деміена Каца (Damien Kats), засновника проєкту Couch DB. Із оглядом усіх цих мов можна ознайомитися в [771]. Запропоновано чимало графових моделей баз даних, огляд яких наведено в [772].

Документно – орієнтовані бази даних.

Документно – орієнтовані бази даних (ДОБД) (document – oriented database) – це база даних, призначена для зберігання й маніпулювання документами. Документ – це деревовидний направлений граф із позначеними вершинами. Листові вершини представляють дані документа, а позначки (імена) решти вершин представляють властивості (атрибути) відповідних даних документа. Документи можуть об′єднуватися у колекції в певному сенсі однотипних документів, але водночас ніякі вимоги щодо однаковості складу атрибутів документів можуть і не ставитися. Колекції можуть містити інші колекції.

Документна структура належить до класу так званих безсхемних (schemaless)

самоописуваних (selfdescribing) структур. Як правило, в документній структурі відсутній розподіл на схему й екземпляр. Схеми немає, а всі необхідні структурні елементи схеми присутні в екземплярі, й у цьому сенсі дані є самоописуваними. Документно – орієнтовані дані також дістали назву слабоструктурованих / напівструктурованих (semistructured data).

Слабоструктуровані дані. Слабоструктурована модель даних заснована на ідеї представлення даних без явного і окремого визначення їхньої схеми. Натомість окремі фрагменти інформації перемежовуються структурними / семантичними тегами, що визначають їхню структуру, вкладеність та інші характеристики. Таке представлення забезпечує гнучкішу обробку й обмін даними.

Термін «слабоструктуровані» дані ввів Лунєвський та інші (Luniewski) 1993 року в системі Rufus [775]. 1995 року Папаконстантину та інші (Papaconstantinou) визначив модель для слабо структурованих даних OEM у рамках системи інтеграції гетерогенних баз даних TSIMMIS [776, 777]. 1996 року Бунеман та інші (Buneman) визначив модель слабоструктурованих даних [778]. 1999 року Дойч та інші (Deutch) описав зв′язок між слабо структурованими даними й XML [780].

Мови запитів. Для слабоструктурованих даних були визначені мови запитів, які дозволяють добувати дані з цієї структури або перетворювати одну слабо структуровану структуру в іншу. Ці мови з′явилися практично одночасно із самою структурою. 1995 року була розроблена система і мова Lorel [781] – одна з перших мов запитів для слабоструктурованих даних, де було введено поняття регулярного вираження шляху для навігації шляхами з частково відомою структурою.В мові UnQL [779] увага акцентується на трансформаціях запитів і вводиться структурна ракурсія як центральна парадигма перетворення слабоструктурованих даних. Мова XML – QL [780] стала першою мовою, де принципи мов слабо структурованих даних були застосовані до XML.

З точки зору внутрішньої структури представлення документна структура є різновидом структури NoSQL ключ – значення, коли значення, в свою чергу, може бути парою «ключ – значення» тощо, представляючи таким чином багаторівневу ієрархію. Для форматування документних даних використовуються стандартні мови JSON, BSON, XML, YAML та інші.

Основні операції практично всіх ДОБД позначаються абревіатурою CRUD і означають Create, Retrieve, Update, Delete.

Припускається, що в будь-якій ДОБД існує механізм задавання унікальних ідентифікаторів документів, за якими відбувається індексація, що суттєво прискорює пошук документів.

Системи ДОБД. Була розроблена чимала кількість систем ДОБД. Так на сайті [783] наведено короткий опис понад 60 систем ДОБД, серед яких на 2022 рік кращими вважаються: Amazon Dynamo DB, Mongo DB, Mongo DB Atlas, Couchbase Server, Google Cloud Firestore, Percona Server for Mongo DB, Inter System IRIS, Asango DB, Database Management, Azure Cosmos DB.

Колончаті бази даних.

Колончата NoSQL база даних (КБД) – це така база даних, де дані зберігаються згуртованими по колонках таблиці, а не рядками, як у реляційних базах даних. У ній «сусідами» є не дані з двох стовпчиків того самого рядка, а дані одного і того ж стовпчика, але з різних рядків.

Зазначимо, що термін «колончата БД» має два значення. (1) *Колончато – орієнтована* – це БД, не обов′язково NoSQL, яка зберігає дані таблиці не рядками, а стовпчиками. Такі системи зазвичай використовуються в аналітичних інструментальних засобах, зокрема, HPE Vertica. (2) *Сімейство колонок* (column family) або *широка колонка* (wide-column) представляють тип NoSQL баз даних, які підтримують таблиці, що мають різну кількість, імена й формати/типи колонок у різних рядках таблиці. В даному розділі мова піде про колончаті БД другого типу.

Колончаті сімейства можуть складатися практично з необмеженої кількості колонок, які можуть створюватися динамічно. Читання й записування відбуваються з використанням колонок, а не рядків.

Колонка може бути представлена у вигляді великої кількості пар ключ – значення, де ключ - ім'я колонки. Таким чином успадковуються властивості сховищ типу ключ – значення.

У деяких випадках розрізняють колончаті сховища й сховища сімейства колонок. У першому випадку мається на увазі, що кожна колонка зберігається самостійно, не залежно від інших колонок, а в другому – всі колонки сімейства запам'ятовуються разом.

Супер – колонка – це колонка, що складається з інших колонок. До прикладу, супер – колонкою є ПІБ, яка складається з колонок Прізвище, Ім'я та по-Батькові. З іншого боку, Прізвище, Ім'я та по-Батькові – це сімейство колонок. Отож, сімейство супер – колонок – це сімейство, складене з сімейств колонок. Така вкладеність може бути багаторазовою. З точки зору концепції сховищ ключ – значення ми маємо ситуацію, коли значення в свою чергу є парою ключ – значення. З іншого боку, користуючись термінологією реляційних баз даних, можна сказати, що сімейство супер – колонок – це в певному сенсі аналог поняття «погляд» (view).

Історія КДБ

Етап 1. Транспоновані файли (1969 – 1985). Вважається, що історія колончатих баз даних починається з кінця 60-х років із появою так званих транспонованих файлів (transposed files), в яких рядки табличних даних переводяться у стовпчики, а стовпчики – у рядки. Першою підтримуючою транспоновані файли СУБД вважається TAXIR (1969) [784], орієнтована на зберігання й пошук біологічних даних. До цього класу належить також розроблена 1975 року медична система TOD [785] і система RAPID [786], створена 1976 року компанією Statistics Canada для пошуку й обробки статистичних даних. Згодом вона використовувалась у багатьох статистичних організаціях до кінця 90-х років. Варто також згадати про створену 1977 року SCSS – колончатий варіант системи SPSS (Statistical Package for the Social Sciences) – статистичний пакет для соціальних наук [787]. Однією з найбільш ранніх систем, що мала риси сучасних колончатих СУБД, вважається Cantor [788]. Були здійснені дослідження з організації пошуку в транспонованих файлах [789]. 1975 року в праці [790] досліджено питання декомпозиції записів на підзаписи з наступним їх зберіганням в окремих файлах.

Етап 2. Модель декомпозованої пам'яті – DSM (1985 – 2000). Наступний етап пов'язаний із появою методів вертикального розбиття, що передбачає поатрибутну кластеризацію таблиць. До цього в базах даних панувала так звана модель N-арної пам'яті NSM (N-ary Storage Model). Але 1985 року була опублікована стаття [791], де як альтернатива NSM була запропонована модель декомпозованої пам'яті (Decomposition Storage Model, DSM). У DSM кожна колонка таблиці запам'ятовується окремо, а щоб знати, якому саме рядку таблиці належить значення в колонці, разом із цим значенням запам'ятовується унікальний ідентифікатор рядка. Здебільшого це сурогатний первинний ключ (дивись рис. нижче).

ID	Товар	Ціна	Дата
1	ПК	500	07.10.21
2	Монітор	150	07.10.21
3	Мишка	15	09.10.21
4	Прінтер	170	11.11.21

а) Модель пам'яті /NSM

	Товар
1	ПК
2	Монітор
3	Мишка
4	Прінтер

	Ціна
1	500
2	150
3	15
4	170

	Дата
1	07.10.21
2	07.10.21
3	09.10.21
4	11.11.21

б) Модель пам'яті DSM

Протягом наступних 20 років використовувались саме терміни NSM/DSM для зазначення рядковості й по колонкової моделей пам′яті. Наступні дві статті, присвячені дослідженню проблеми розпаралелювання операцій роботи з DSM [792], а також використанню індексів для виконання операцій з′єднання й проєкції [793], продемонстрували безперечну перевагу DSM перед NSM під час виконання операцій вибірки даних із БД.

Етап 3. Бурхливий розвиток (2000+). До початку 2000-х років концепція DSM не була затребувана, бо були відсутні класи задач, котрі гостро потребували КБД. І лише з появою статистичних і аналітичних баз даних, сховищ даних, технологій OLAP і великих даних, КБД стали доволі затребуваними. Першими дослідницькими прототипами КБД, реалізованими в 2000 – 2005 рр., які значно вплинули на подальший розвиток комерційних КБД, були MonetDB [794, 795], VectorWise (MonetDB/X 100) [796] и C-Store [798].

Першими комерційними КБД вважаються Sybase IQ (1996) і KDB (1998). Друга половина 2000-х років відзначилася бурхливим ростом кількості колончатих СУБД. У цей час були реалізовані Vertica (Vertica Analytic Database),, Exasol, ParAccel, Kognito, InfoBright, SAND, Ingres VectorWise, Kickfire, Paraccel. Прикладами колончатих баз даних є також Apache Cassandra, Scylla, Apache HBase, Google BigTable, Microsoft Azure Cosmos DB. Колончато – орієнтовані засоби впроваджені в такі реляційні СУБД, як Oracle, SQLServer, PostgreSQL, IBM BLU.

На початку 2000-х років виникла ідея гібридних сховищ, що підтримують як рядкові, так і колончаті зберігання даних. Так 2001 року було розроблено сховище PAX (Partition Attributes Across), 2002-го запропоновано гібридну модель “Fractured Mirrors”. Пізніше були реалізовані комерційні гібридні сховища SAP HANA, Infini DB, Greenplum.

На момент написання статті в базі даних https://dbdb.io/ (Database of Databases) була представлена 51 СУБД, що мала відношення до колончатої моделі даних.

Характерні риси і сфера застосування. Характерними рисами КБД є:

- висока швидкість виконання операцій пошуку/доступу, особливо під час виконання запитів з агрегатними функціями;
- висока горизонтальна масштабність завдяки повній свободі у розподілення колонок між вузлами мережі;
- висока ефективність декомпозиції й стискання даних;
- можливість функціонування у безсхемному варіанті.

Колончаті СУБД знаходять застосування переважно в аналітичних схемах класу business intelligence, аналітичних OLAP – сховищах даних (data warehouses) і системах класу Big Data.

Методи реалізації й оптимізація. Хоча сучасні комерційні КБД широко використовують принципи й методи, запропоновані для транспонованих файлів і декомпозованих структур, однак вони мають можливості, які на початкових етапах не були передбачені. Особливо це стосується питань, пов′язаних із аспектами оптимізації і підвищення продуктивності їх функціонування. Наведемо їх.

Віртуальний ключ. Якщо значення колонки є фіксованої довжини, то можна не зберігати сурогатний ключ разом із кожним значенням колонки, а обчислювати як значення ключа, так і розміщення відповідного значення колонки на основі зміщення. Такий принцип був використаний у Monet DB [795]. Ним можна користуватися у разі відсутності сортування в колонках.

Блокова організація й векторна обробка. Для КБД були запропоновані методи блокової організації даних та їх векторної обробки [796, 800], які суттєво підвищують їхню продуктивність. Блокова ітерація [797] передбачає, що велика кількість значень колонки передається у вигляді одного блоку від одного оператора до наступного.

Пізня матеріалізація (late materialization). Пізня матеріалізація

або пізнє відтворення кортежа, потрібного для запиту, має на увазі максимально можливе відтермінування виконання операцій з′єднання колонок [801, 802]. Пізня матеріалізація значно підвищує ефективність використання пропускної здатності пам′яті.

Стискання колонок. Стискання колонки з використанням найефективнішого для нього методу дає суттєве зменшення розмірів файлів колонок [800, 803]. Через те, що колонки містять дані одного типу (атрибуту), досягаються хороші показники стискання за допомогою простих алгоритмів. Було запропоновано багато алгоритмів стискання для колончатих сховищ [804].

Оперування стиснутими даними. У багатьох сучасних колончатих сховищах розпакування даних відтермінovується, допоки це не буде абсолютно необхідним [800, 801]. В ідеальному варіанті – поки не з′явиться необхідність представити результати користувачеві. В зв′язку з цим вирішується задача оперування стиснутими даними. Пізня матеріалізація дозволяє колонкам залишатися стиснутими доти, доки не з′явиться потреба формувати з них кортежі (виробляти їх з′єднання).

Ефективна реалізація операції з′єднання. Оскільки КБД передбачають колончате представлення даних, важливим є питання використання різних стратегій виконання операції з′єднання. В КБД використовуються як класичні алгоритми, так і специфічні [800, 805, 806].

Надлишкове представлення окремих стовпців із різним сортуванням. По колонках, відсортованих відносно конкретного атрибуту, можна здійснювати швидкий пошук. Збереження кількох копій конкретної колонки, відсортованих за різними атрибутами, може суттєво підвищити продуктивність виконання запитів. До прикладу, система C – Store [798] фізично зберігає колекції колонок, кожна з яких відсортована за певним атрибутом. Групи колонок, відсортовані за певним атрибутом, називаються проєкціями. Одна й та ж колонка може знаходитися в різних проєкціях. Наявність різних сортувань сприяє оптимізації функціонування системи.

Крекінг і адаптивне індексування баз даних. Крекінг (Cracking) – це принципово новий підхід у базах даних, який базується на принципі, що створення й ведення індексу є продуктом діяльності з обробки запитів, а не створення й оновлення бази даних. Кожен запит інтерпретується не лише як запит на отримання частки бази даних, а й як вказівка на необхідність «відколювання» від неї невеликих шматочків, описуваних цим запитом. Із цих шматочків практично будується крек – індекс, аби збільшити швидкість подальшого пошуку. Крек – індекс створюється й підтримується динамічно залежно від обробки запитів і адаптується з урахуванням зміни робочих навантажень запитів. Щодо КБД загальна схема функціонування механізму крекінгу наступна [809]. Щоразу, як запит уперше формулюється щодо атрибуту А, механізм крекінгу створює копію колонки атрибуту А, яка називається крекінг – колонкою А. В процесі виконання подальших запитів розпізнаються ті, які звертаються до атрибуту А, й відбувається подальше настроювання крекінг – колонки А та її індексу. Більше того, крекінг – колонка А використовується для збільшення швидкості подальшого пошуку за атрибутом А. Monet DB була однією з перших КБД, яка підтримувала механізм крекінгу. Детальніше ознайомитися з механізмами крекінгу й адаптивного індексування можна в монографії [804].

Ефективне завантаження й оновлення. У зв′язку з тим, що в КБД дані декомпозовані по колонках і активно використовується механізм стискання даних, завантаження й оновлення БД відбувається значно повільніше, ніж у рядкових БД. Тому були досліджені питання оптимізації виконання цих функцій [798, 810]. Так, зокрема, в S-Store [798] дані спочатку записуються в оптимізований для запису буфер, що нести скається. А потім періодично «скидаються» у великі пакети, які стискаються.

Циклотрон даних. Одним із головних завдань розподіленої обробки запитів є розробка самоорганізованої архітектури, здатної оптимально використовувати всі апаратні ресурси для оперативного управління базою даних. Це необхідно для мінімізації часу відгуку на запити й максимізації пропускної здатності за відсутності єдиної глобальної точки координації. Було запропоновано технологію *Циклотрона даних* (Data Cyclotron) [804a], яка вирішує цю проблему використанням турбулентного переміщення даних через кільце зберігання, створене з розподіленої оперативної пам′яті з використанням функціональних можливостей, які пропонуються сучасними мережевими засобами віддаленого прямого доступу до пам′яті (Remote Direct Memory Access – RDMA). Запити, ініційовані в окремих вузлах мережі, постійно взаємодіють із кільцем зберігання, збираючи фрагменти даних, які постійно циркулюють у кільці.

Здійснено порівняльні дослідження рядкових і колончатих сховищ [806 - 808]. Зокрема, результати досліджень [806] показали, що оптимізований варіант колончатого сховища працює вп’ятеро швидше за комерційні рядкові сховища.

За адресою https://www.predictiveanalyticstoday.com/top-wide-columnar-store-databases/ можна ознайомитися із такими 9 найпопулярнішими КБД 2021 року згідно із PAT Research: MariaDB, CrateDB, ClickHouse, Greenplum Database, Apache Hbase, Apache Kudu, Apache Parquet, Hypertable, MonetDB.

Так само відповідно до сайту https://www.g2.com/ наступні 7 КБД були найпопулярнішими 2021 року: Amazon Redshift, Snowflake, ClickHouse, Druid, Hbase, Apache Kudu, Apache Parquet (https://www.g2.com/categories/columnar-databases).

Для глибшого ознайомлення із КБД рекомендуємо монографію [804], а також статті [811 - 813]. Чудовим навчальним матеріалом є посібник [814].

Завершимо виклад колончатих СУБД таким висловом, узятим із [804]: «Сучасні колончаті сховища вийшли за межі простого поколончатого зберігання даних, вони пропонують цілком нову архітектуру баз даних і механізми роботи з ними, адаптовані для сучасних технічних засобів і методів аналітичної обробки даних».

Графові бази даних.

Графова база даних (ГБД) – це БД, у якої модель даних має графову структуру певного виду, включно зі схемою та її екземплярами, а маніпулювання даними здійснюється за допомогою мов, що мають граф – орієнтовані оператори.

Взаємозв′язок між графовою структурою і базами даних відслідковується від моменту виникнення баз даних у 60-і роки минулого століття. Структура даних ієрархічних і мережевих баз даних, а також ER-мови, нагадують графову структуру, однак вони не є ГБД.

ГБД виявляються доволі корисними, або й незамінними у випадку, коли інформація про взаємозв′язок даних є важливішою або настільки ж важливою, як і власне дані. В такому разі дані й взаємозв′язки між ними перебувають на одному логічному рівні.

Згідно з [823a] виділяються дві категорії ГБД: *транзакційні* й *нетранзакційні* ГБД. Перші мають відношення до великої сукупності невеликих графів, як-от лінгвістичні дерева. Характерними операціями для них є пошук суперграфів, підграфів, подібних графів. До другої категорії належать спільні великі ГБД, наприклад, соціальні мережі, які можуть складатися з кількох компонент. Характерні операції – пошук (найкоротшого) шляху, пошук сусідів, пошук компонент із заданими властивостями.

Можна виділити два етапи досліджень і розробок в галузі ГБД. На першому етапі (80-і – початок 90-х років) активно проводилися дослідження у сфері моделей даних і мов запитів ГБД. Пізніше інтерес до них суттєво знизився через появу геопросторових, темпоральних, слабоструктурованих і XML – баз

даних. Однак на початку цього століття ці дослідження і розробки поновилися в зв′язку з появою семантичного ВЕБу, зв′язаних даних і соціальних мереж. Цей тип характеризується передусім тим, що крім досліджень було розроблено багато промислових ГБД.

Графові моделі даних. У ГБД не існує канонічної моделі даних, як-от, скажімо, в реляційній моделі. Це пояснюється гнучкістю графової структури, яка дозволяє нарощувати її складність, а також визначає різні мови маніпулювання і запитів. З огляду на це було запропоновано чимало моделей і мов запитів ГБД. Усе ж виділяють такі типи моделей графових баз даних [825].

Базова графова модель – мічені графи. Складається з вершин (вузлів) і направлених дуг (ребер), котрі з′єднують вершини. Вершини представляють концепти (об′єкти), а дуги - зв′язки між цими концептами. Графова структура застосовується для завдання як схеми БД, так і її екземплярів. Вершини/дуги мають унікальні ідентифікатори й нуль або більше міток (labels) [835, 839, 840, 848]. Зазвичай міткам даються імена відповідних концептів, а дугам – імена відповідних зв′язків. Мітки дають змогу вказати класи, до яких належать вершини/дуги. Складніший варіант базової структури – мультограф, у якому пара вершин може бути зв′язана кількома дугами.

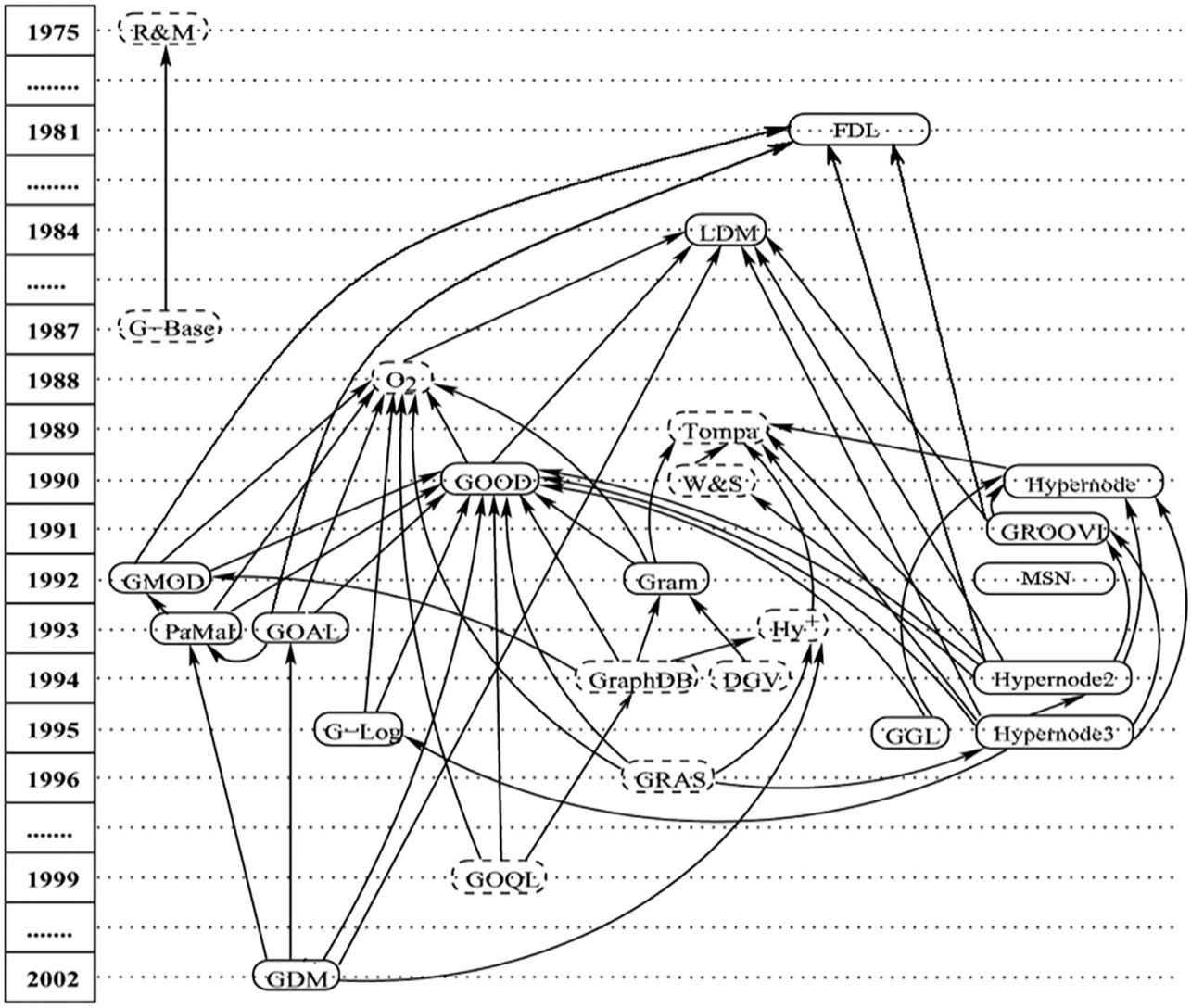
Графова модель із властивостями (Property Graph Model). Граф із властивостями – це спрямований мічений атрибутивний мультиграф. Поняття графу з властивостями введене в [827], а його формальне визначення дано в [824]. Вершини/дуги можуть мати багато властивостей (атрибутів), які дозволяють вказати їхні характеристики. Властивості задаються у вигляді пар ключ – значення. Мовами графових моделей із властивостями є G-CORE [823], PROPER [854], Gremlin [885], Cypher [887], PGQL [891].

Гіперграфова модель – складні дуги. Гіперграф (hipergraph) – це узагальнення графової моделі даних, в якій дуги можуть з′єднувати будь-яку кількість вершин, як на початку дуги, так і в її кінці [828]. Такі дуги називаються гіпердугами (hiperedge). Гіперграфи виявляються корисними, коли дані містять зв′язки типу «багато до багатьох». Гіперграфові моделі представлені в працях [834, 836, 838, 844].

Модель гіпервершин – вкладені графи. Модель гіпервершин (hipernodes) – це спрямований граф, у якому вершини самі можуть бути графами. Такі вершини називаються гіпервершинами, утворюючи структуру вкладених графів (nested graphs). Модель дозволяє наочно і природно представляти об′єкти довільної складності. Модель гіпервершин уперше була визначена 1990 року в [837], а згодом уточнена цими ж авторами в [847]. Даній моделі присвячені також статті [838, 849, 850]. Аналогічна ідея запропонована в мультимасштабних мережах (multi – scaled networks) [841].

Граф даних веба – модель RDF. RDF – це мова представлення взаємопов′язаних ресурсів у вебі з використанням графічної структури даних триплетів, у якій використовуються спрямовані дуги й мічені вершини та дуги. Використовувана в RDF графічна структура є найбільш загальною в тому сенсі, що в ній дуги є також вершинами. Це дозволяє підтримувати принцип самоописуваності (реїфікації), тобто формувати твердження відносно тверджень. У RDF – графі одночасно присутні схема і її екземпляри, для відокремлення екземплярів від схем використовуються мічені спеціальним іменем дуги (ім′я – type). Модель RDF власну мову запитів SPARQL. До баз даних, що підтримують модель даних RDF, належать: 4Store, AllegroGraph, BigData, Jena TDB, Sesame, Stardog, OWLIM, uRiK.

У праці [819] подається детальний огляд графових моделей даних до 2003 року. На рисунку нижче, взятому з [819], представлені найвідоміші графові моделі баз даних, впорядковані за роками їх публікації. Тут овали представляють моделі, стрілки вказують на цитування, а пунктирні овали свідчать про те, що відповідні роботи мають відношення до графової моделі БД.



Далі наводиться короткий опис кожної з моделей із зазначенням статей їх публікації.

- R&M [829] – введено поняття семантичної мережі для зберігання інформації про дані бази даних.
- FDL [830] – у мережах функціональної моделі даних неявно визначена графова структура даних, метою якої було забезпечення «концептуального природного» інтерфейса бази даних.
- LDM [831] – у межах логічної моделі даних явно визначена графова модель бази даних, мета якої – узагальнення реляційної, ієрархічної й мережевої моделей даних.
- G-Base[832] – запропонована графова модель даних, названа G-Base, для представлення складних структур.
- O2 [833] – визначена об′єктно – орієнтована модель даних O2 на основі графової структури.
- Tompa [834] – модель даних для гіпертекстових баз даних.
- GOOD [835] – граф – орієнтована об′єктна модель, призначена для систем із графо подібними засобами представлення і маніпулювання даними.
- W&S [836] – гіперграфова модель для доступу до даних у базі даних.
- Hypernode [837] – граф – орієнтована модель із гіпервершинами, що являють собою вкладені графи.
- GROOVY [838] - об′єктно – орієнтована гіперграфова модель даних.
- GMOD [839] – висунуто ряд концептуальних пропозицій щодо інтерфейсів користувача в граф – орієнтованій базі даних.

- Gram [840] – граф – орієнтована модель для представлення гіпертекстів.
- MSN [841] – графова модель мультимасштабних мереж, яка об′єднує основи теорії графів і об′єктно – орієнтовану парадигму.
- PaMal [842] – є розширенням GOOD з явним представленням картежів і множин.
- GOAL [843] – граф – орієнтована об′єктна модель із асоціативними вершинами.
- Hy+ [844] – гіперграфова модель з мовами запитів і візуалізації.
- Graph DB [845] – графова модель даних і мова запитів для баз даних.
- DGV [846] – графова модель для визначення і маніпулювання графами різного виду, які зберігаються в реляційних або об′єктних базах даних.
- Hypernode2 [847] – модель із вкладеними графами, яка є розвитком Hypernode.
- G-Log [848] – граф – орієнтована модель і декларативна мова запитів.
- GGL [849] – теоретико-графова модель даних баз даних карт генома.
- Hypernode3 [850] – графова модель даних, яка є розвитком Hypernode.
- GRAS [851] – графова атрибутивна модель для представлення складної інформації.
- GOQL [852] - об′єктно-орієнтована графова модель даних і графова мова запитів.
- GDM [853] – граф-орієнтована модель із n-арними семитричними зв′язками.

Графові мови запитів (ГМЗ).

Модель ГБД, окрім структури і обмеженої цілісності, має високорівневу графову мову запитів (ГМЗ), в якій можна формулювати специфічні для графів операції. В [819] подається огляд ГМЗ періоду першої хвилі досліджень і розробок у царині ГБД. На підставі цієї праці в статті [855] було досліджено питання виразної потужності й обчислювальної складності деяких із цих мов. У статті [821] подається порівняльний аналіз мов, функціонуючих на той час ГБД. У статті [856] досліджується проблема формулювання запитів до ГБД і, зокрема, виразність і складність навігаційних мов запитів. І, нарешті, в праці [822] подано огляд основоположних особливостей сучасних ГМЗ, а в [825] наводиться аналітичний огляд ГМЗ станом на 2018 рік.

Як уже згадувалося, ГМЗ мають специфічні для графів операції. Коротко опишемо їх.

Суміжність (adjacency) і окіл (neighborhood). Дві вершини суміжні, якщо вони з′єднуються дугою, дві дуги суміжні, якщо вони мають спільну вершину. В статті [857] досліджується питання ефективності виконання операції суміжності у великих динамічних розряджених графах. Більш загальним поняттям є «окіл». N – околом заданої вершини є множинність вершин, які є досяжними із заданої вершини за допомогою шляху, що охоплює не більше n – дуг. Дослідженню проблеми суміжності/околу в базах даних присвячена монографія [858].

Співставлення зі зразками (pattern matching). Знаходження множини під графів заданого графа бази даних, які відповідають заданому графу – зразку. Завдання пошуку за зразком характерне для багатьох класів прикладних завдань. Скажімо, розпізнавання образів, ідентифікація співтовариств у соціальних мережах. Проблема співставлення зі зразком досліджується в теорії баз даних [861, 862], біоінформатиці [863], семантичному вебі [864]. Як показано в [859, 860] проблема співставлення зі зразком має відношення до проблеми інтелектуального аналізу графів даних (data graph mining).

*Досяжність / зв'язність (reachability / connectivity).*Проблема досяжності полягає у встановленні, чи існує шлях, що веде від однієї вершини до другої. В цьому контексті розрізняють два види шляхів: шляхи фіксованої довжини

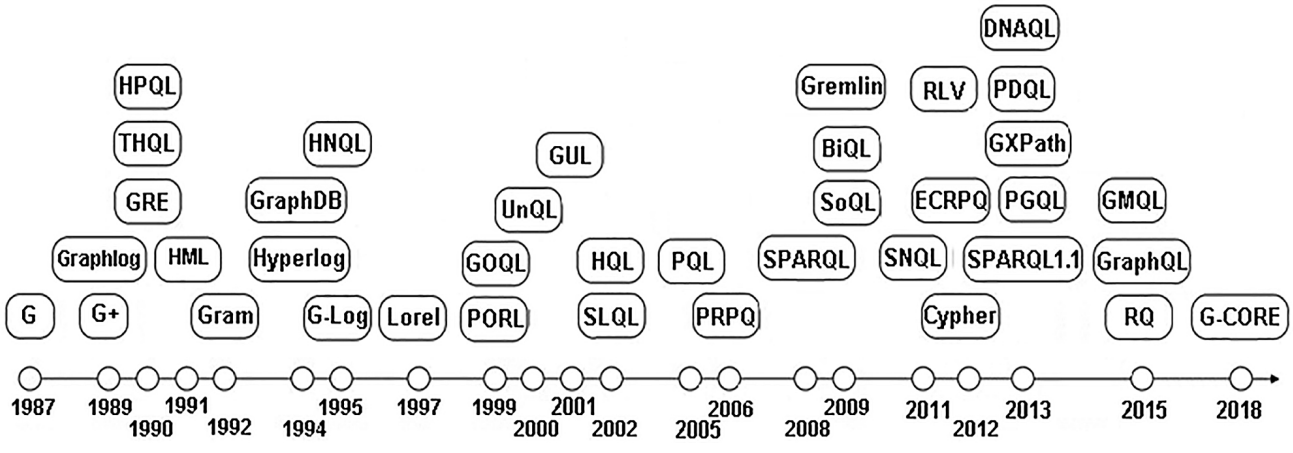
(fixed length paths), які містять фіксовану кількість вершин і дуг, а також регулярні прості шляхи (regular simple paths), в яких накладаються обмеження (регулярні вирази) на вершини / дуги. В статті [861] подано огляд різних теоретико – графових задач, пов′язаних зі шляхами, які мають відношення до баз даних разом із обчисленням транзитивного замикання, виконанням рекурсивних запитів і складного пошуку шляхів включно. В [865] вводиться поняття запиту з регулярним шляхом (regular path query – RPQ) як способу вираження запитів на досяжність, а в [866] аналізується проблема досяжності з урахуванням наявності регулярних виразів. Цей тип запитів досліджується також у працях [856, 867]. Одним із різновидів задачі досяжності є знаходження найкоротшого шляху, якщо їх, скажімо, кілька [868].

Аналітичні запити. Агрегування. Запити цього виду не працюють зі структурного графу, а надають кількісну інформацію (здебільшого в агрегатному вигляді) щодо топологічних властивостей графа бази даних. Аналітичні запити зазвичай містять спеціальні агрегатні оператори типу count, sum, min, max, average, які підсумовують результати запиту у вигляді кількості вершин, кількості сусідів, довжини шляху, відстані між вершинами, найкоротшого шляху між вершинами тощо. Як показано в монографії [859], складні аналітичні запити тісно пов′язані з алгоритмами інтелектуального аналізу графа даних.

Приблизне співставлення і ранжування (approximate matching and ranking). Можливі ситуації, коли користувачі не знають структуру графа, щодо якого формулюються запити, а також, існуючі в ньому обмеження і правила. Як наслідок, вони можуть формулювати запити, котрі не даватимуть результати, або ж відповіді не відповідатимуть очікуваним результатам. У такому разі бажано мати можливості отримання неточних результатів і їх ранжування відповідно до встановлених критеріїв. Однією з перших праць, щодо формулювання «гнучких» запитів до слабо структурованих текстів, була [869], де досліджується питання неточного відпрацювання запиту на основі спеціального виду відповідності між запитом і графом. У праці [870] дається більш узагальнене поняття приблизного співставлення шляхів графа, коли результати пошуку можуть бути ранжовані згідно із їхньою «близькістю» до вихідного запиту. На рисунку нижче, який є дещо зміненим і доповненим варіантом із праці [825], наведені в хронологічному порядку так звані «чисті» ГМЗ, тобто ті, що призначені для роботи з графічними моделями даних.

Ці мови описуються в наступних статтях:

G [865], G+ [871], Graphlog [872], HPQL [837], THQL [836], GRE [873], HML [838] Gram [840], Hyperlog [847], GraphDB [845], G−Log [848], HNQL [850], Lorel [874], PORL [875], GOQL [852], UnQL [876], GUL [877], SLQL [878], HQL [879], PQL [880], PRPQ [881], SPARQL [882], SoQL [883], BiQL [884], Gremlin [885], SNQL [886] Cypher [887], ECRPQ [888], RLV [889], SPARQL 1.1



[890], PGQL [891], GXPath [892], PDQL [826], DNAQL [893], RQ [894]. GraphQL [895], GMQL [896], G-CORE [823].

Для ГБД є принциповим ефективне виконання запитів [897, 898]. З цією метою було розроблено різні методи їх індексування [899 - 901]й оптимізації [902 - 904].

Графові бази даних (ГБД). За останні 20 років було реалізовано понад 60 графових баз даних. Їх списки присутні на багатьох сайтах.

https://hostingdata.co.uk/nosql-database/, на цьому сайті, присвяченому NoSQL – базам даних, є розділ зі списком графових баз даних з адресами в Інтернеті, за якими можна з ними ознайомитися.

На сайті https://dbdb.io/, який являє собою «базу даних баз даних» і містить інформацію про понад 760 СУБД, наведений перелік понад 40 систем БД, що базуються на графовій моделі даних (https://dbdb.io/browse?data-model=graph) із зазначенням основних відомостей і адрес веб – сайтів, за якими можна з ними ознайомитися.

https://sourceforge.net/software/graph-databases/?page=1 за цією адресою наводяться відомості щодо 50 графових систем баз даних, адреси веб – сайтів, на яких можна з ними ознайомитися.

Відзначимо, що в усіх трьох джерелах дати реалізації систем наведені після 2000 року.

У списку нижче поданий узагальнений перелік ГБД у хронологічному вигляді.

Існує велика кількість статей із порівняльним аналізом різних ГБД, як-от: [817, 821, 905 - 907].

Нижче наведено таблицю порівняльного аналізу 20 ГБД взяту з [817].

На закінчення підкреслимо великий внесок у розвиток теорії й методології графових баз даних доктора кафедри комп'ютерних наук Університету Талька (Чилі) Ренцо Англеса (Renzo Angles). Даний розділ написано здебільшого за матеріалами його праць.

Ренцо Англес

2002	InfinityDB
2005	AllegroGraph
2006	Blazegraph, Sparksee
2007	DEX, Neo4J, HyperGraphDB, AnzoGraph, sones GraphDB
2008	InfoGrid
2009	VertexDB, Pregel, SylvaDB, IBM System G
2010	HyperGraphDB, InfiniteGraph, Sones, OrientDB, FlockDB, Filament, G-Store, Redis_graph, Horton, CloudGraph, Stig
2011	ArangoDB, Trinity, OrigoDB, ArangoDB, Fallen-8
2012	GraphChi-DB, GrapheneDB, SparkleDB, Sqrrl, TigerGraph, FaunaDB, JanusGraph
2013	Bitsy, imGraph, AgensGraph, Galaxybase
2014	Cayley, GraphDB, GrapheekDB, GUN
2015	Cosmos DB, DegDB, DGraph
2016	IndraDB, EliasDB, Memgraph, VertexDB, TypeDB
2017	JanusGraph, Amazon Neptune, Fluree
2018	AnzoGraph DB, Nebula Graph, Neptune
2019	TerminusDB

NewSQL – бази даних

2007 року Майкл Стоунбрейкер, розробник систем баз даних Ingres і Postgres, майбутній лауреат премії Тьюрінга, очолив дослідницьку групу, яка опублікувала основоположну статтю [908], де підкреслювалося, що апаратні припущення в основі реляційної архітектури більше не придатні. Стоунбрейкер і його команда запропонували ряд варіантів перспективних проєктних рішень щодо СУБД, два з яких стали особливо важливими для подальшого розвитку напрямку, який згодом дістав назву NewSQL. Це H-Store [909] – розподілена база даних, яка повністю міститься в пам′яті, та C-Store [910] – колончата база даних. 2010 року Рік Кеттелл (Rick Cattell) опублікував статтю [911], де використав термін «масштабований SQL», і проаналізував такі відомі на той час масштабовані реляційні бази даних, як MySQL, Cluster, VoltDB, Clustrix, ScaleDB, ScaleBase, NimbusDB, а також порівняв підходи масштабованої SQL і NoSQL.

Термін NewSQL запропонований 2011 року аналітиком 451 Group Метью Аслетом (Matthew Aslett) [912, 913]. Відтоді він застосовується на означення масштабованих реляційних систем управління базами даних нового покоління з оперативною обробкою транзакцій (OLTP), які мають здатність горизонтальної масштабованості NoSQL і підтримки ACID, характерної для традиційних (SQL) систем баз даних.

Метью Аслет

Пропоновано також варіанти класифікації NewSQL – баз даних [915, 916].

Пізніше були здійснені дослідження, в результаті яких з′явилося чимало публікацій щодо порівняльного аналізу технологій SQL, NoSQL та NewSQL. Фундаментальною працею в цьому напрямку стала монографія Гая Харрісона (Guy Harrison) [12].

На сьогодні NewSQL – технологія знайшла свою нішу на ринку баз даних і широко використовується в промисловості. До цього класу належать наступні системи: MemSQL, VoltDB, Spanner, Calvin, CockroachDB, FaunaDB, YugabyteDB.

Гай Харрісон

Онтологічні бази даних

Із появою семантичного вебу з′явилося поняття онтології. Було висунуто кілька визначень онтологій. Найповніше й загальноприйняте наступне [917]:

Онтологія – це очевидна формальна специфікація погодженої концептуалізації. Детальніше про це визначення в [918].

Сьогодні онтології широко використовуються в різних галузях. Вони стали важливою складовою семантичного вебу. Розроблено інструменти маніпулювання онтологіями, як-от Protege. Однак вони не надають тих можливостей, які дають БД, і, передовсім, постійне зберігання, маніпулювання й формулювання запитів до структури онтології та її даних. У зв′язку з цим виникло по-

няття **онтологічної бази даних (ОнБД).** Це база даних, яка дає можливість зберігати й маніпулювати онтологіями, включно з онтологічною структурою і даними цієї структури. В контексті семантичного вебу було запропоновано декілька підходів створення ОнБД [919 - 922]. З їх оглядом можна ознайомитися в [923]. Найчастіше ОнБД створюються переважно на базі реляційних баз даних. Окрім того, мовою представлення онтологій нижнього рівня обирається RDF.

Пропонується декілька моделей ОнБД, найпопулярніші з яких наведені далі.

Безсхемна модель (schema-oblivious), названа також вертикальною (vertical). В цій моделі онтологія зберігається в єдиній тернарній таблиці у вигляді RDF-триплетів < суб′єкт – предикат - об′єкт>. Ця таблиця містить і структуру онтології, і її дані. Модель представлена в Jena [924, 925], 3store [922], Rstar [926], Virtuoso [927], Oracle [928]. Суттєва перевага – простота підтримки моделі.

Схемна модель (schema-aware) також має назву бінарної (binary). Кожен клас і кожна властивість онтології (RDF/S-схеми) мають власну таблицю [920, 921, 929, 930]. Класи містяться в унарних таблицях, а властивості – в бінарних. Таблиця властивості об′єднує індивіди різних класів, що мають ці властивості. Перевага цієї моделі – підтримка багатозначних властивостей. Модель використовується в системі управління даними SOR IBM [931].

Розширеним варіантом схемної моделі є дуальна (dual) модель, яка може містити не лише схеми класів і властивостей, а й схеми (мета-схеми) структури онтології. Одним із варіантів є мета-схеми IS-A включення одних класів або властивостей в інші, описуючи таким чином таксономічну ієрархію класів і властивостей. У варіанті ISA-схеми задається таксономічна таблиця для класів/властивостей, екземплярами якої є пари класів/властивостей, що перебувають у цьому відношенні.

Крім того, мета-схема може включати завдання області визначення (domaim) і області значення (range) властивостей. Такий підхід застосовується в ОнБД OntoDB [932].

Гібридна модель (hybrid) [929] поєднує властивості двох попередніх. Використовується тернарна таблиця для кожного типу області значення властивості й бінарна таблиця для всіх екземплярів усіх класів.

Горизонтальна модель. Онтологія представляється у вигляді реляційних таблиць, коли властивості класу стають атрибутами таблиці класу. Якщо ж властивість є багатозначною, то вона представляється бінарною таблицею. За умови, що всі властивості багатозначні, то ця модель перетворюється в бінарну. Дана модель використовується в OntoMS [933], OntoDB [932] и Jena2 [934], Існують два різновиди цієї моделі:

- *Таблиця кластеризації за властивостями* (clustered property table): виділяється група властивостей і створюється таблиця із усіма індивідами (екземплярами) онтології, які мають ці властивості не залежно від їх приналежності до класу. Тобто таблиця може містити індивіди різних класів.

- *Таблиця властивість – клас* (property-class table): таблиця має всі індивіди одного класу із заданим набором властивостей. Одна й та ж властивість може бути у різних таблицях.

В обох різновидах за умови існування індивідів, що не потрапляють у жодну із цих таблиць, вони розміщуються у вертикальні таблиці.

Усі ці моделі ОнБД були реалізовані в існуючих RDF-сховищах (RDFSuite, Jena, Sesame, DLDB, RStar, KAON, PARKA, 3Store, Oracle), вичерпний огляд яких подано в [935].

Виведення. В онтологіях існує проблема виведення, коли за таксономічною ієрархією включення класів/властивостей необхідно створити їх транзитивне замикання. В ОнБД пропонується два підходи вирішення цієї задачі: а) або попередньо вираховувати й матеріалізувати їх (під час компіляції) – цей підхід названо MatView і використовується у без схемному підході; б) або вирахову-

вати їх за необхідності (під час виконання). Другий варіант використовується у схемному й гібридному підході.

Мови. Була запропонована велика кількість мов вебу й семантичного вебу, огляд яких наведено в статті [936]. Серед них виділяється клас мов, що працюють із форматом RDF. До них належать: мови сімейства SPARQL (SquishQL, RDQL, SPARQL, TriQL.), мови сімейства RQL (RQL, SeRQL, eRQL), мови з реактивними правилами (Algae, iTQL, WQL), дедуктивні мови запитів (N3QL, R-DEVICE, TRIPLE, Xcerpt). Усі вони так чи інакше можуть бути застосовані для ОнБД, які базуються на RDF. Водночас в [937] описується мова OntoQL, розроблена для ОнБД OntoDB. Крім того, в цій статті визначена алгебра онтологій, на базі якої створена мова OntoQL.

Повнотекстові бази даних

Повнотекстова база даних (ПТБД) – це база даних, що містить повнотекстові документи і уможливлює їх відшукувати. Документи можуть містити звичайний текст, структурований, слабо структурований або неструктурований. Може мати спеціальні описуючі елементи, так звані метадані, а також мати мультимедійні компоненти.

Історично проблемою зберігання текстових документів у комп′ютерах почали перейматися практично одночасно із питанням зберігання числових даних. Перші документальні системи давали можливість редагувати й форматувати тексти й були частиною видавничих систем. Електронний документообіг став невід′ємною частиною автоматизації діловодства, яка почала активно розвиватися в 1970-х і 1980-х роках. Зберігання повнотекстових документів у комп′ютерах спричинило появу гіпертекстової та гіпермедійної технології, що нині є ядром всесвітньої павутини.

Однією з перших систем управління повнотекстовими документами була STAIRS (STorage And Information Retrieval System – система зберігання й пошуку інформації), розроблена в IBM 1969 року для власних внутрішніх потреб. А 1973 року була представлена як комерційний продукт. Однак ПТБД почали широко використовуватися лише на початку 90-х років, коли комп′ютерні технології зберігання зробили їх економічно вигідними й технологічно можливими. Існує два основні класи ПТБД: розширення класичних бібліографічних баз даних до ПТБД і ПТБД на основі Інтернету (на основі пошукових систем чи XML).

Метадані. В ПТБД повнотекстові документи можуть містити описи, такі, як автори, назва, анотації, ключові слова й УДК для наукових статей. Такі описи дістали назву метаданих. Різні типи інформаційних ресурсів можуть мати різні набори метаданих, які дістали назву схем метаданих. До прикладу, розроблені схеми метаданих для опису персон і організацій (vCard і FOAF), бібліографічних ресурсів (MARC, UNIMARC, DC), музичних та історичних цінностей (CDWA), архівів і електронних ресурсів (GILS, EAD) та багато інших. В електронних бібліотеках найбільш використовуваною є схема Дублінського ядра (Dublin Core).

Індексація. За умови невеликої кількості документів незначного розміру під час пошуку здійснюється послідовний огляд усього документа. Але за умови великих розмірів послідовний перегляд неефективний. Тому в цьому випадку попередньо відбувається індексування – побудова списку пошукових термінів із зазначенням місць їхньої зустрічі. В загальному випадку пошуковими термінами стають усі слова тексту документа. Крім тексту власне документа індексації можуть піддаватися метадані. Для забезпечення витонченіших пошукових можливостей процес індексації може включати додаткові можливості, як-от:

- не враховувати незначущі слова такі, як займенники , прийменники, сполучники, вигуки тощо;
- окрім місця знаходження слова запам′ятовувати його порядкове місце в тексті (пошук поблизу слів у фразі);

- разом зі словом запам′ятовувати його контекст, тобто – фрагмент тексту, де воно знаходиться;

- всі однокореневі слова наводяться в індексі лише раз у вигляді одного нормованого слова.

Мови запитів. Мови запитів сучасних пошукових систем в ПТБД дають розвинуті засоби точнішого формулювання того, що саме треба знайти, включають:

- специфікацію простору пошуку (тобто підмножини ПТБД), в якому слід здійснювати пошук;

- використання логічних виразів;

- специфікацію як окремих пошукових термінів, так і фраз;

- специфікацію відстані між словами, на якій вони перебувають в документі;

- використання регулярних виразів;

- специфікацію поточного пошуку, тобто знаходження документів, які містять слова, в певному сенсі близькі до вказаних, у пошуковому виразі.

Всі ці й багато інших можливостей реалізуються як за допомогою розвинутих засобів індексації, згаданих вище, так і за допомогою розробки сучасних механізмів пошуку, огляд яких наводиться далі.

У подальшому ми будемо використовувати наступні терміни, усталені в публікаціях за алгоритмами пошуку:

- **зразок** (sample) – термін або фраза, що шукаються;

- **текст** (text) – документ, в якому здійснюється пошук;

- **співставлення** (matching) – процедура знаходження зразка в тексті.

Існують різні способи класифікації алгоритмів співставлення, наприклад:

- точне/неточне співставлення;

- співставлення зліва направо, справа наліво у вказаному порядку, в довільному порядку;

- з урахуванням або без урахування стоп – слів;

- знаходження в тексті одного чи всіх зразків;

і багато інших прикладів. За основу ми беремо перший варіант, а в межах кожного із цих класів будемо наводити їх різновиди.

Алгоритми точного співставлення

Вони передбачають точну відповідність тексту зразкові. Відомі наступні п′ять варіантів: символьний, з хешируванням, автоматний, біт – паралельний, гібридний.

Символьний підхід. Символьний підхід є класичним підходом співставлення рядків, який передбачає по символьне порівняння зразка з текстом.

Найпростішим варіантом є повний перебір (brute-force). Порівняння зразка починається від самого початку тексту. Порядок посимвольного порівняння не визначається. У разі невдалого порівняння зразок зсувається на одну символьну позицію праворуч, і процес порівняння повторюється. Від моменту його появи було докладено чимало зусиль із розробки ефективніших алгоритмів, основні з яких коротко описані нижче.

Алгоритм Кнута – Морріса – Пратта ***(Knuth–Morris–Pratt – KMP)*** – ефективний алгоритм, здійснюючий пошук підрядка в рядку. Час роботи алгоритму лінійно залежить від об′єму вхідних даних. Спочатку алгоритм був запропонований 1970 року Джеймсом Моррісом (James H. Morris) і Воганом Праттом (Vaughan Pratt) [946] й незалежно від них досліджений Дональдом Кнутом (Donald Knuth) [947]. Зрештою ці троє вчених опублікували спільну статтю 1977 року [948]. Незалежно від них 1971 року радянський вчений Юрій Матіясевич запропонував аналогічний алгоритм під час дослідження завдання співставлення рядків у двійковому алфавіті [949].

Алгоритм Бойєра – Мура ***(Boyer-Moore – BM).*** 1977 року було висунуто пропозицію алгоритму співставлення Бойєра – Мура [950]. Розроблений Робертом Бойєром (Robert S. Boyer) і Джеєм Муром (J Strother Moore), він є алгоритмом загального призначення і став стандартним й еталонним алгоритмом цього класу. Алгоритм передбачає дві фази:

- фаза попередньої обробки. Згідно із певним правилом створюється таблиця, що визначає величини, на які слід робити зсуви зразка праворуч у разі його невдалого порівняння з текстом.

- Фаза порівняння. Зразок порівнюється із текстом справа наліво. В процесі порівняння вираховується значення зсуву за замовчуванням.

Алгоритм ВМ є найефективнішим алгоритмом пошуку для багатьох додатків, зокрема, текстових редакторів. Він добре працює для алфавітів помірного розміру і довгих шаблонів. Проте розміри алфавіту й зразків суттєво впливають на час попередньої обробки. Алгоритм ВМ докладно описано в [952]. Згодом з'явилося багато алгоритмів, які або модифікують ВМ, або об'єднують його з іншими підходами. До модифікуючих варіантів ВМ, поліпшуючих деякі його характеристики, належать алгоритми Хорспула (Horspool) 186], Чжу-Такаока (Zhu-Takaoka) [954], Turbo-BM [955], Апостоліко - Джанкарло (Apostolico and Giancarlo) [957] Сміта (Smith) [958], Райта (Raita) [959], Крочемора [960], Беррі-Равіндрана [961]. Так само гібридні ВМ – підходи передбачають інтеграцію ВМ – алгоритму з іншими методами для підвищення продуктивності. Вони описані в статтях [962 - 968].

Символьний підхід найбільше придатний для додатків, де шукаються великі текстові зразки.

Підхід із хешируванням. Хеш – підхід передбачає попередньо перед порівнянням застосувати хеш – функцію до зразка й порівнюваного рядка, аби порівнювати не символи, а числа для підвищення швидкості. Цей підхід був розроблений 1987 року Майклом Рабіном (Michael O. Rabin) и Річардом Карпом (Richard M. Karp).[969]. Передбачає дві фази:

- попередня обробка – це однократне обчислення хеш – функції зразка й багатократне обчислення хеш – функцій порівняльних підрядків тексту;

- порівняння хеш – функцій зразка й підрядка. У разі їх рівності проводиться додаткове посилювальне порівняння зразка із підрядком через те, що хеш – функція може видавати однакові числа для різних рядків.

Проблема даного підходу полягає в перерахуванні хешу для кожного підрядка. А її вирішення – у використанні для підрахунку наступного хеш – значення поточного хешу. Це досягається шляхом використання так званого кільцевого хешу. Робін і Карп запропонували використовувати поліноміальний хеш – різновид кільцевого.

Хороший опис алгоритму Робіна – Карпа наведений у статті [970]. Було висунуто ще кілька алгоритмів пошуку з хешируванням, описані в [971 - 973].

Оригінальним різновидом хеш – підходу є так званий q-gram – підхід, який передбачає розбиття зразка на частини, обчислення хеш – функцій кожної частини і подальше їх використання у порівнянні зразка із підрядком тексту. За приклад можуть бути алгоритми, описані в [974, 975].

Цей підхід найвдаліше використовується там, де необхідна швидкість співставлення рядків.

Автоматний підхід. Автоматний підхід у вирішенні задачі співставлення рядків передбачає використання концепції автомату. Одним із варіантів є так званий підхід із використанням суфіксного автомату. Він використовує два взаємозв'язані, але різні автомати: детермінований ациклічний кінцевий автомат (deterministic acyclic finite state automaton) [976], який представляє кінцеву множину рядків, і суфіксний автомат, виконуючий функцію суфіксного індексу [977].

Суфіксний автомат – це найменший частковий детермінований кінцевий автомат, який розпізнає набір суфіксів даного рядка. Граф станів суфіксного автомату має назву «орієнтований ациклічний граф слів» (directed acyclic word graph - DAWG). Суфіксний автомат уперше був описаний групою вчених із Денверського і Колорадського університетів 1983 року [978]. Існує три різновиди автоматичного підходу, котрі коротко подані далі.

1) Орієнтований ациклічний граф слів (DAWG). DAWG – це структура даних, яка забезпечує швидкий пошук слів. У DAWG вершина представляє символ. Спеціальна вершина представляє початковий символ. Частина вершин є кінцевими. Вершини мають спрямовані

зв′язки. Переміщення від початкової вершини до інших уздовж зв′язків вирішує задачу співставлення. Були досліджені такі різновиди DAWG-співставлень: зворотній не детермінований (backward non-deterministic) DAWG [979], подвійний прямий (double-forward) DAWG [980], алгоритм співставлення на основі зворотнього прогнозу (Backward oracle matching - BOM) [981] і його різновидів [982, 983].

2) Широке вікно. Широке вікно – це вікно, розмір якого перевищує розмір зразка, через що в результаті переміщення зразка в процесі пошуку вікна перекриваються, а це дає певні переваги. Вперше ідея широких вікон була висунута 2005 року в праці [984]. Ця особливість широких вікон враховується під час використання суфіксного автомату. З часом цей алгоритм був удосконалений у працях [985, 986]. Цей підхід зараховують також до класу біт паралельних алгоритмів.

3) Автоматний підхід із пропуском непотрібних спроб співставлення. Автоматний підхід до співставлення темпоральних (тобто залежних від часу) зразків із можливим пропуском непотрібних спроб співставлення був запропонований у праці [987]. Темпоральний підхід до проблеми співставлення характерний для систем реального часу і веб – додатків.

Біт – паралельний підхід. Побітові операції над комп′ютерними словами типу NOT, OR, AND, XOR мають властивий їм паралелізм. Вважається [988, 993], що бітовий алгоритм точного пошуку винайдений угорським ученим Балінтом Дьомьолкі (Domolki) [989? 990] 1964 року й розширений Шьямасундаром (Shyamasundar) [991] 1977 року перш, ніж він був заново винайдений Рікардо Баеза-Йейтсом и Гастоном Гонне (Ricardo Baeza-Yates and Gaston Gonnet) 1992 року [992] і названий Shift OR. Цей алгоритм з часом був розширений і удосконалений [994, 998].

Після Shift OR 1998 року було запропоновано біт – паралельний алгоритм BNDM (Backward Non Deterministic Matching) [999, 1000], що належить до класу Shift AND алгоритмів. Тобто використовуються операції Shift і AND для паралельного пошуку. Згодом були висунуті удосконалені варіанти BNDM: TNDM (Two way Non Deterministic Matching) [1001], SBNDM (Simplified BNDM) [1001], BNDMq и SBNDMq (q-gram BNDM/SBNDM) [1002], MBNDMq (Multiple BNDMq) [1003], LBNDM (long BNDM) [1001], FBNDM (Forward BNDM) [1004].

Одним із різновидів біт – паралельного пошуку є апаратний підхід, який передбачає використання комп′ютерної SIMD-архітектури. SIMD-алгоритми розроблені для пришвидшення виконання співставлення рядків із використанням апаратного підходу. А саме – функціональних можливостей SIMD-команд. До них належать алгоритми, описані в [1005 - 1007]. Вдалий огляд біт – паралельних алгоритмів наведено у праці [1008].

Цей підхід особливо швидкий, коли зразок не перевищує розмір комп′ютерного слова.

Гібридний підхід. Він добре підходить для вирішення важких задач, адже поєднує переваги різних алгоритмів. Багато алгоритмів використовують гібридну концепцію. Скажімо, алгоритми, описані в [1009 - 1011] поєднують символьний і автоматний підходи, а [1012 - 1013] – символьний і хешированій.

Алгоритми неточного співставлення

Пропонувалися і досліджувалися також алгоритми неточного (приблизного) співставлення.

Неточні алгоритми передбачають введення поняття відстані між рядками і знаходження тих, що містяться на відстані, яка не перевищує задану відносно вихідного рядка.

В зв′язку з цим було введене поняття відстані реагування як способу кількісної оцінки того, наскільки два рядки (не) схожі один на одного, через підрахунок мінімальної кількості операцій, необхідних для перетворення одного рядка в інший.

У статті «String metric» у Вікіпедії (https://en.wikipedia.org/wiki/String_metric) представлені 20 видів метрик на рядках. Наведемо ті з методів визначення ступеню схожості, які найбільше згадувані в науковій літературі.

Відстань Хеммінга (Hamming distance). Введена Річардом Хеммінгом (Richard Hamming) [1014] 1950 року, визначає число позицій, в яких відповідні символи двох слів однакової довжини є відмінними. В більш загальному випадку відстань Хеммінга застосовується для рядків однакової довжини і є метрикою відмінності об′єктів однакової розмірності.

Відстань Левенштейна. Введена радянським ученим Владіміром Левенштейном [1015, 1016] 1965 року і визначається для рядків різної довжини як кількість односимвольних операцій вставки, видалення і заміни, що переводять один рядок в інший. Внесок у вивчення цієї відстані зробив також Ден Гасфілд (Dan Gusfield) [1017].

Відстань Дамерау – Левенштейна (Damerau–Levenshtein). Є узагальненням відстані Левенштейна, запропонована Фрідеріком Дамерау (Frederick J. Damerau) [1018]. Це міра різниці двох рядків символів, що визначається як мінімальна кількість операцій вставки, видалення, зміни й транспозиції (перестановки двох сусідніх символів), необхідних для переведення одного рядка в інший.

Відстань Джаро – Вінклера (Jaro–Winkler distance). Подана 1990 року Уільямом Е. Вінклером (William E. Winkler) [1019] на основі відстані Джаро (Matthew A. Jaro) [1020], запропонованої 1989 року. Неформально відстань Джаро між двома словами – це мінімальна кількість односимвольних перестановок, необхідна для зміни одного слова в інше.

Найбільша загальна послідовність (longest common subsequence) [1021] визначає відстань із використанням операцій вставки й видалення (без заміни).

На основі перелічених вище відстаней була запропонована велика кількість алгоритмів неточного співставлення рядків, із якими можна ознайомитися в оглядах [1022 - 1028].

Відстань при впорядкованому алфавіті. Всі перелічені вище алгоритми неточного співставлення рядків не враховують факт можливого впорядкування символів алфавіту. В свою чергу впорядкований алфавіт дозволяє вводити відстань між символами алфавіту, між двома рядками й на їх основі – відповідні методи співставлення рядків. Так, зокрема, в праці[937a] визначаються Δ-відстань, Г-відстань і (Δ; Г)-відстань між рядками й відповідні методи неточного співставлення рядків.

Огляд літератури. На закінчення огляду алгоритмів співставлення наведемо ще кілька статей.

Вдалий огляд точного співставлення рядків дано в [939]. У статті подана широка класифікація методів співставлення, опис понад 50 алгоритмів і їх порівняльний аналіз. Обговорюються нові напрямки, можливі проблеми, поточні тенденції в сфері алгоритмів співставлення рядків з основним наголосом на алгоритми точного співставлення.

У монографії французьких учених Крістіан Чаррас (Christian Charras) і Тьєррі Лекрок (Tierry Lecrock) [941] подано детальний опис 35 алгоритмів точного співставлення рядків.

Достойний цикл статей представив італійський учений Сімоне Фаро (Simone Faro) та його колеги. В статті [942] наводиться класифікація алгоритмів точного співставлення рядків і 124 алгоритми зі стислою анотацією, які пропонувалися протягом 1970 – 2016 років, з наведенням бібліографії. В праці [943] описується ефективний і гнучкий інструментальний засіб SMART (String Matching Algorithms Research Tool), призначений для розробки, тестування, порівняння і оцінки алгоритмів співставлення рядків. Наводиться також список 124 алгоритмів співставлення, розроблених за період 1970 – 2016 років і представлених в SMART. У [944] наводиться огляд алгоритмів точного співставлення рядків, розроблених після 2000 року, експериментальні дані оцінки деяких із цих алгоритмів.

Електронні бібліотеки

Електронна бібліотека – це розподілена документальна інформаційно – пошукова система, створена на базі ПТБД, яка надає різнорідні колекції електронних документів у глобальній мережі комп'ютерів у зручному для користувачів вигляді.

Однією з ранніх праць на тему електронних бібліотек була монографія Ліклайдера (Joseph Carl Robnett Licklider) [1029], написана 1965 року. В ній передбачалося існування всесвітньої мережі комп'ютерів, яка містить оцифровані версії будь – коли написаної літератури, до якої надається безпосередній доступ.

Джозеф Ліклайдер

До середини 70-х років здійснювалися дослідження й розробки у сфері онлайнових інформаційних систем і сервісів із ЕБ включно. Їх детальний аналіз здійснено у монографії [1030]. Серед перших проєктів, які можна віднести до ЕБ, вважається Проєкт «Гутенберг» [1031] – громадська некомерційна ініціатива, спрямована на створення й розповсюдження цифрової колекції творів, які є суспільним надбанням. Проєкт було створено 4 липня 1971 року, коли студент Іллінойського університету Майкл Харт (Michael Stern Hart) вручну передрукував текст Декларації незалежності США і відправив його іншим користувачам своєї мережі. Станом на 2021 рік у колекції Проєкту було понад 60000 книг.

Майкл Харт

Термін «електронна бібліотека» (digital library) почав широко використовуватися з 1991 року в зв′язку із проведенням серії семінарів, фінансованих Національним науковим фондом США(US National Science Foundation NSF). Того ж року була створена система “e-print archive”, яку згодом перейменували на arXiv.

У середині 90-х була усвідомлена важливість створення ЕБ на державних рівнях. 1997 року в США стартував національний проєкт «Ініціатива електронних бібліотек» (Digital Library Initiative - DLI), що став серйозним поштовхом у розвитку як ЕБ, так і національних програм, проєктів і організаційних структур в інших країнах: Великобританії, Канаді, Австралії, Японії, ЄС тощо. 1996 року в спеціальному випуску журналу EEE Computer [1032], присвяченому побудові великих ЕБ, підбивалися підсумки виконання проєктів аж до 1996 року.

У зв′язку із активним зростанням кількості ЕБ до кінця минулого століття наукова громадськість дійшла розуміння необхідності створення концептуальних положень і моделей ЕБ вцілому і для різних предметних галузей зокрема. Тож у

наступному десятилітті було висунуто багато таких моделей.

- У 1991 – 1997 роках Міжнародною федерацією бібліотечних асоціацій і закладів була розроблена ER-модель «Функціональні вимоги до бібліографічних записів» (Functional Requirements for Bibliographic Records, FRBR) [1033] як узагальнене представлення бібліографічного універсаму, незалежного від будь-якого варіанту каталогізації або реалізації.

- Концептуальна еталонна модель CIDOC CRM [1034] Міжнародного комітету із документації Міжнародної ради музеїв, призначена для інтеграції, посередництва і обміну інформацією в сфері світової культурної спадщини й пов′язаних сфер.

- 1991 року був представлений Загальноєвропейський дослідницький інформаційний формат CERIF (Common European Research Information Format - CERIF) [1035] як всеохоплюючої інформаційної моделі предметної галузі наукових досліджень. Модель CERIF є стандартом в ЄС.

- Вже багато років досліджується проблема аналізу семантики зв′язків між науковими матеріалами. Системним узагальненням цих результатів став комплекс антологій SPAR (Semantic Publishing and Referencing) [1036], який забезпечував доволі детальну категоризацію відношень, що можуть виникати між науковими матеріалами в електронному вигляді, і втілюючих їх зв′язків.

- Група спеціалістів DELOS Європейського дослідного консорціуму з інформатики й математики ERCIM у 2006 – 2007 роках на основі аналізу бібліотечних систем [1037], де велику увагу приділялося функціональним можливостям сучасних ЕБ, спочатку сформулювали маніфест ЕБ [1038]. У ньому були висвітлені наріжні концептуальні принципи ЕБ, а потім розроблено еталонну модель ЕБ DLRM (Digital Library Reference Model) [1039], яка стала де-факто стандартом у Європейському Союзі.

- Гонсалвес (Goncalves) та інші запропонували модель 5S [1040, 1041] як теоретичної бази ЕБ, що сприяло появі великої кількості мета моделей для різних типів ЕБ.

Поява тисяч ЕБ в Інтернеті викликала необхідність їх інтеграції. Серед цілої низки пропозицій у цьому напрямку найефективнішим й загальноприйнятим став протокол OAI PMH – протокол міжбібліотечного обміну метаданими, створений 2001 року [1042]. Його створення привело до появи сайтів – інтеграторів (харвесторів), які об′єднують велику кількість ЕБ й надають спільну точку пошуку й доступу до них. Прикладами таких харвесторів можуть бути наступні:

- Bielefeld University Library - http://www.base-search.net/ - пошукова система Білефельдського університету інтегрує ресурси понад 9000 провайдерів даних;

OpenAire – інформаційні ресурси відкритого доступу Європейського Союзу (56 мільйонів документів) - https://www.openaire.eu/;

Core – найбільша в світі колекція (понад 135 мільйонів) наукових статей відкритого доступу **-** https://core.ac.uk/.

Було розроблено понад 30 інструментальних засобів створення ЕБ (див.: http://roar.eprints.org/). Найпопулярнішими є DSpace, EPrints, OJS, Bepress, OPUS, Fedora, Greenstone.

References

755. Strozzi C. NoSQL – A relational database management system. 2007–2010. – http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page
756. Evans E. NoSQL 2009. May 2009. – Blog post of 2009-05-12. - http://blog.sym-link.com/posts/2009/12/nosql_2009/
757. Evans E. NoSQL: What's in a name? October 2009. – Blog post of 2009-10-30. - http://blog.sym-link.com/posts/2009/30/nosql_whats_in_a_name/
758. Fox A, Brewer E. Harvest, yield and scalable tolerant systems. In: Proceedings of Workshop on Hot Topics in Operating Systems; 1999. p. 174–178.
759. Seth Gilbert, Nancy Lynch. Brewer's conjecture and the feasibility of consistent,

available, partition-tolerant web services. ACM SIGACT News, Volume 33 Issue 2, June 2002, pp. 51-59,

760. Abadi D. Consistency tradeoffs in modern distributed database system design: CAP is only part of the story. Computer 45(2), 37-42 (2012)
761. Strauch Ch. "NoSQL Databases". - http://www.christof-strauch.de/ nosqldbs.pdf
762. Kepner J., Chaidez J., Gadepally, Jansen H. "Associative arrays: Unified mathematics for spreadsheets, databases, matrices, and graphs," New England Database Day, 2015.
763. Kepner J., Chaidez J., "The Abstract Algebra of Big Data and Associative Arrays," SIAM Meeting on Discrete Math, Jun 2014, Minneapolis, MN.
764. Jeremy Kepner, Vijay Gadepally, Dylan Hutchison, Hayden Jananthan, Timothy Mattson, Siddharth Samsi, Albert Reuther. Associative Array Model of SQL, NoSQL, and NewSQL Databases. 2016 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–9. IEEE (2016)
765. A Brief History of NoSQL. - http://blog.knuthaugen.no/2010/03/a-brief-history-of-nosql.html
766. GT.M - https://en.wikipedia.org/wiki/GT.M
767. DB-Engines Ranking of Key-value Stores. - db-engines.com/en/ranking/key-value+store
768. Rusher J., Networks R. Triple Store. - https://www.w3.org/2001/sw/Europe/events/20031113-storage/positions/rusher.html
769. Tweed R., James G. A Universal NoSQL Engine, Using a Tried and Tested Technology. - http://www.mgateway.com/docs/universalNoSQL.pdf, 2010. - 25 p.
770. Welcome to the UnQL Specification home - http://www.unqlspec.org/display/UnQL
771. Bach M., Werner A. Standardization of NoSQL Database Languages. In: Kozielski S., Mrozek D., Kasprowski P., Małysiak-Mrozek B., Kostrzewa D. (eds) Beyond Databases, Architectures, and Structures. BDAS 2014. Communications in Computer and Information Science, vol 424. Springer, Cham. 2014, pp. 50–60
772. Angles R., Gutierrez C. Survey of graph database models. ACM Comput. Surv. 40, 1, Article 1, 2008, 39 p.
773. Suciu D. Semi-structured Data Model. In Encyclopedia of Database Sys-tems, Ling Liu, M. Tamer Özsu Editors, pp. 3446-3451
774. Suciu D. Semi-structured Query Languages. In Encyclopedia of Database Sys-tems, Ling Liu, M. Tamer Özsu Editors, pp. 3457-3459
775. Luniewski A., Shoens K., Schwarz P., Stamos J., Thomas J. The Rufus system: information organization for semi-structured data. In: Proceedings of the 19th International Conference on Very Large Data Bases; 1993. p. 97–107.
776. Papakonstantinou Y., Garcia-Molina H., Widom J. Object exchange across heterogeneous information sources. In: Proceedings of the 11th International Conference on Data Engineering; 1995. p. 251–260.
777. Garcia-Molina H., Papakonstantinou Y., Quass D., Rajaraman A., Sagiv Y., Ullman J., Widom J. The TSIMMIS project: integration of heterogeneous information sources. J Intell Inf Syst. 1997;8(2):117–132.
778. Buneman P., Davidson S., Hillebrand G., Suciu D. A query language and optimization techniques for unstructured data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 1996. p. 505–516.
779. Buneman P., Fernandez M., Suciu D. UNQL: a query language and algebra for semistructured data based on structural recursion. VLDB J. 2000;9(1): 76–110.
780. Deutsch A., Fernandez M., Florescu D., Levy A., Suciu D. A query language for XML. In: Proceedings of the 8th International World Wide Web Conference; 1999. p. 77–91.
781. Abiteboul S., Quass D., McHugh J., Widom J., Wiener J. The Lorel query language for semistructured data. 1996. http://www-db.stanford.edu/lore/.
782. Papakonstantinou Y., Abiteboul S., Garcia-Molina H. Object fusion in mediator systems. In: Proceedings of the 22th International Conference on Very Large Data Bases; 1996. p. 413–424.

783. Best Document Databases. - https://www.g2.com/categories/document-databases
784. Estabrook G F., Brill R.C. The Theory of the TAXIR accessioner. Mathematical Biosciences, 1969, Vol. 5, No 3–4, pp. 327-340.
785. Weyl S. Fries J.F. Wiederhold G., Germano F."A Modular Self-describing Clinical Databank System". Computers and Biomedical Research. 1975. 8 (3): 279–293.
786. Turner M.J., Hammond R., Cotton P. A DBMS for Large Statistical Databases. VLDB '79: Proceedings of the fifth international conference on Very Large Data Bases - Vol. 5, 1979, pp. 319–327
787. "SCSS from SPSS, Inc". ComputerWorld. September 26, 1977. p. 28.
788. Karasalo I., Svensson P. An overview of cantor: a new system for data analysis. SSDBM'83: Proceedings of the Second International Workshop on Statistical Database ManagementSeptember, 1983, pp.315–324
789. Don S. Batory. On searching transposed files. ACM Transactions on Database Systems, 4(4):531–544, 1979.
790. Hoffer J.A. , Severance D.G. The use of cluster analysis in physical data base design. In VLDB '75: Proceedings of the 1st International Conference on Very Large Data BasesSeptember 1975 Pages 69–86, 1975.
791. Copeland G.P., Khoshafian S.N. . A decomposition storage model. In Proceedings of the ACM SIGMOD Conference on Management of Data,1985, pp. 268–279
792. Khoshafian S., Valduriez P. Parallel execution strategies for declustered databases. In Proceedings of the International Workshop on Database Machines, pages 458–471, 1987.
793. Khoshafian S., Copeland G., Jagodis T., Boral H., Valduriez P. A query processing strategy for the decomposed storage model. In Proceedings of the International Conference on Data Endineering (ICDE), pp. 636–643, 1987.
794. Boncz P. Monet: A next-generation DBMS kernel for queryintensive applications. University of Amsterdam, PhD Thesis, 2002.
795. Idreos S., Groffen F., Nes N., Manegold S., Mullender S., Kersten M.L MonetDB: Two Decades of Research in Column-oriented Database Architectures. IEEE Data Eng. Bull., 35(1):40–45, 2012.
796. Boncz P., Zukowski M., Nes N. MonetDB/X100: Hyperpipelining query execution. In Proceedings of the biennial Conference on Innovative Data Systems Research (CIDR), 2005, pp. 225-.237
797. Zukowski M., Boncz P.A., Nes N, Heman S. MonetDB/X100 - A DBMS In The CPU Cache. IEEE Data Engineering Bulletin, 28(2): 17–22, June 2005.
798. Michael Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Samuel R. Madden, Elizabeth J. O'Neil, Patrick E. O'Neil, Alexander Rasin, Nga Tran, and Stan B. Zdonik. C-Store: A Column-Oriented DBMS. In Proceedings of the International Conference on Very Large Data Bases (VLDB), pages 553–564, 2005.
799. Lamb A., Fuller M., Varadarajan R., Tran N., Vandiver B., Doshi L., Bear C. The Vertica analytic database: C-store 7 years later. Proceedings of the VLDB Endowment, Vol. 5, No 12, 2012 pp 1790–1801.
800. Abadi D.J., Madden S.R., Ferreira M. Integrating compression and execution in column-oriented database systems. In Proceedings of the ACM SIGMOD Conference on Management of Data, pp. 671–682, 2006.
801. Abadi D.J., Myers D.S., DeWitt D.J., Madden S.R. Materialization strategies in a column-oriented DBMS. In Proceedings of the International Conference on Data Endineering (ICDE), pp. 466–475, 2007.
802. Idreos S., Kersten M.L., Manegold S. Self-organizing tuple reconstruction in column stores. In Proceedings of the ACM SIGMOD Conference on Management of Data, pp. 297–308, 2009.
803. Zukowski M., Heman S., Nes N., Boncz P. Super-Scalar RAM-CPU Cache Compression. In Proceedings of the 22nd International Conference on Data Endineering (ICDE), 2006. pp. 59-71

804. Abadi D.J., Boncz P., Harizopoulos S., Idreos S., Madden S. (2013), "The Design and Implementation of Modern Column-Oriented Database Systems", Foundations and Trends® in Databases: Vol. 5: No. 3, pp 197-280.

804a. Goncalves R., Kersten M.L. The Data Cyclotron Query Processing Scheme. ACM Transactions on Database Systems, Vo. 36. No 4. December 2011, Article No. 27, pp. 1–35

805. Manegold S., Boncz P., Nes N., Kersten M.. Cache-conscious radixdecluster projections. In Proceedings of the International Conference on Very Large Data Bases (VLDB), pages 684–695, 2004.

806. 22) Abadi D.J., Madden S.R., Hachem N. Column-stores vs. row-stores: how different are they really? In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 2008. p. 967– 980.

807. Halverson A., Beckmann J.L., Naughton J.F., DeWitt D.J. A Comparison of C-Store and Row-Store in a Common Framework. Technical Report TR1570, University of Wisconsin-Madison, 2006. - https://minds.wisconsin.edu/bitstream/handle/1793/60514/TR1570.pdf?sequence=1

808. Harizopoulos S., Liang V., Abadi D.J., Madden S.R. Performance tradeoffs in read-optimized databases. In VLDB, pages 487–498, 2006.

809. Idreos S., Kersten M., Manegold S. Database Cracking. Conference: CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, 2007, pp. 68-78

810. Héman S., Zukowski M., Nes N.J., Sidirourgos L., Boncz P. Positional update handling in column stores. In Proceedings of the ACM SIGMOD Conference on Management of Data, pp. 543–554, 2010.

811. Pingpeng Yuan and Hai Jin. Column Stores. In: Encyclopedia of Database Systems, Ling Liu, M. Tamer Özsu Editors. pp. 518-523.

812. Abadi D.J., Boncz P.A. Harizopoulos S. Column-oriented database systems. Proceedings of the VLDB Endowment, Vol. 2, No. 2, 2009, pp. 1664–1665

813. Kanungo A. Column oriented databases. International Journal of Advanced Computational Engineering and Networking, 2017, Vol. 5, No 8, pp. 10-13

814. Abadi D., Boncz P., Harizopoulos S. VLDB 2009 Tutorial on Column-Stores. - https://www.slideshare.net/abadid/vldb-2009-tutorial-on-columnstores

815. Robinson I., Webber J., Eifrem E. Graph Databases, 2nd Edition. O'Reilly Media, Inc. 2015, 218 p.

816. Wood P.T. Graph Database. In Encyclopedia of Database Systems, Ling Liu, M. Tamer Özsu Editors, pp. 1639-1643

817. Angles R. Graph Databases - http://renzoangles.net/gdm/

818. Angles R., Gutierrez C Querying RDF data from a graph database perspective European semantic web conference, 2005, pp. 346-360

819. Angles R., Gutierrez C. Survey of graph database models. ACM Computing Surveys, Vol. 40, No. 1, Article 1, 2008, pp. 1-39.

820. Angles R., Gutierrez C. The expressive power of SPARQL International Semantic Web Conference, 2008, pp.114-129

821. Angles R. A comparison of current graph database models. IEEE 28th International Conference on Data Engineering Workshops, 2012, 171-177

822. Angles R., Arenas M., Barceló P., Hogan A., Reutter J., Vrgoč D. Foundations of modern query languages for graph databases. ACM Computing Surveys (CSUR), 2017, Vol. 50, No 5, Article No.: 68, pp. 1–40

823. Angles R., Arenas M., Barceló P., Boncz P., Fletcher G., Gutierrez C. G-CORE: A core for future graph query languages. Proceedings of the 2018 International Conference on Management of Data, 2018, pp. 1421-1432

823a. Sakr S. Pardede M. (Eds.). Graph Data Management: Techniques and Applications. IGI Global, 2011, 502 p.

824. Angles R. The property graph database model. In Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, Cali, Colombia, CEUR Workshop Proceedings. CEUR-WS.org, 2018, [Online] URL: http://ceur-ws.org/Vol-2100/paper26.pdf

825. Angles R., Gutierrez C. An Introduction to Graph Data Management: Fundamental Issues and Recent Developments. in Graph Data Management, Springer Publishing Companyt, 2018, pp.1-32
826. Angles R., Barcelo P., Rios G. A practical query language for graph DBs. In: 7th Alberto Mendelzon International Workshop on Foundations of Data Management (AMW), 2013
827. Rodriguez M.A., Neubauer P. Constructions from dots and lines. Bulletin of the American Society for Information Science and Technology, 2010, 36,(6), pp. 35-41
828. Berge C. Graph and Hypergraphs. North-Holland Publishing Company, Amsterdam, 1973
829. Roussopoulos N., Mylopoulos, J. Using semantic networks for database management. In Proceedings of the International Conference on Very Large Data Bases (VLDB). ACM, 1975, 144–172
830. Shipman D.W. The functional data model and the data language DAPLEX. ACM Transactions on Database Systems, vol. 6, No. 1, 1981, pp. 140–173
831. Kuper G.M., Vardi M.Y. A new approach to database logic. In Proceedings of the 3th Symposium on Principles of Database Systems (PODS). ACM Press, 1984, pp. 86–96
832. Kunii H.S. DBMS with graph data model for knowledge handling. In Proceedings of the 1987 Fall Joint Computer Conference on Exploring technology: Today and Tomorrow. IEEE Computer Society Press, 1987, pp. 138–142
833. Lecluse C., Richard P., Velez F. O2, an object-oriented data model. In Proceedings of the ACM SIGMOD International Conference on Management of Data. ACM Press, 1988, pp. 424–433.
834. Tompa F.W. A data model for flexible hypertext database systems. ACM Transactions on Information Systems, Vol. 7, No 1, 1989, pp. 85–100
835. Gyssens M., Paredaens J., Den Bussche J.V., Gucht D.V. A graph-oriented object database model. In Proceedings of the 9th Symposium on Principles of Database Systems (PODS). ACM Press, 1990, pp. 417–424
836. Watters C., Shepherd M.A. A transient hypergraph-based model for data access. ACM Trans. Inform. Syst. 8 (2), 1990, pp. 77–102
837. Levene M., Poulovassilis A. The Hypernode model and its associated query language. In Proceedings of the 5th Jerusalem Conference on Information technology. IEEE Computer Society Press, 1990, pp. 520–530
838. Levene M., Poulovassilis A. An object-oriented data model formalised through hypergraphs. Data Knowl. Eng. 6 (3), 1991, pp. 205–224
839. Andries M., Gemis M., Paredaens J., Thyssens I., Den Bussche J.V. Concepts for graph-oriented object manipulation. In Proceedings of the 3rd International Conference on Extending Database Technology (EDBT). LNCS, vol. 580. Springer, 1992., pp. 21–38
840. Amann B., Scholl M. Gram: A Graph Data Model and Query Language. In European Conference on Hypertext Technology (ECHT). ACM, 1992, pp. 201–211
841. Mainguenaud M., Simatic X.T. A data model to deal with multi-scaled networks. Computers, Environment and Urban Systems, 1992, vol.16, No 4, pp. 281–288
842. Gemis M., Paredaens J. An object-oriented pattern matching language. In Proceedings of the First JSSST International Symposium on Object Technologies for Advanced Software. Springer-Verlag, 1993, pp. 339–355
843. Hidders J., Paredaens J. GOAL, A graph-based object and association language. Advances in Database Systems: Implementations and Applications, CISM, 1993, pp. 247–265
844. Consens M., Mendelzon A. Hy+: A hygraph-based query and visualization system. ACM SIGMOD Record, Vol. 22, No 2, 1993, pp. 511–516
845. Guting R.H. GraphDB: modeling and querying graphs in databases. In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB). Morgan Kaufmann, 1994, pp. 297–308
846. Gutierrez A., Pucheral P., Steffen H., Thevenin J.-M. Database graph views: A practical model to manage persistent

graphs. In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB). Morgan Kaufmann, 1994. pp. 391–402

847. Poulovassilis A., Levene M. A Nested-Graph Model for the Representation and Manipulation of Complex Objects. ACM Transactions on Information Systems (TOIS) 12(1), 1994, pp. 35–68
848. Paredaens J., Peelman P., Tanca L. G-Log: A graph-based query language. IEEE Trans. Knowl. Data Eng. 7, 3, 1995, pp. 436–453
849. Graves M., Bergeman E.R., Lawrence C.B. A graph-theoretic data model for genome mapping databases. In Proceedings of the 28th Hawaii International Conference on System Sciences (HICSS). IEEE Computer Society, 1995, pp. 32-41
850. Levene M., Loizou G. A graph-based data model and its ramifications. IEEE Trans. Knowl. Data Eng. 7, 5, 1995, pp. 809–823
851. KIesel N., Schurr A., Westfechtel B. GRAS, graph-oriented software engineering database system.Information Systems, Vol. 20, No 1, 1995, pp. 21-51
852. Sheng L., Ozsoyoglu Z. M., Ozsoyoglu G. A graph query language and its query processing. In Proceedings of the 15th International Conference on Data Engineering (ICDE). IEEE Computer Society, 1999, pp. 572–581.
853. Hidders J. Typing graph-manipulation operations. In Proceedings of the 9th International Conference on Database Theory (ICDT). Springer-Verlag, 2002. pp. 394–409
854. Spyratos N., Sugibuchi T. (2016) PROPER - A Graph Data Model Based on Property Graphs. In: Grant E., Kotzinos D., Laurent D., Spyratos N., Tanaka Y. (eds) Information Search, Integration, and Personalization. ISIP 2015, pp. 23-35
855. Wood, P.T.: Query languages for graph databases. ACM SIGMOD Record, 2012, Vol. 41, No 1, pp. 50–60.
856. Barceló P. Querying graph databases. In: PODS '13: Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI symposium on Principles of database systems, 2013, pp. 175–188
857. Kowalik L Adjacency queries in dynamic sparse graphs. IInformation Processing Letters, 2007, vol. 102, pp. 191–195
858. Papadopoulos A.N., Manolopoulos Y. Nearest neighbor search - a database perspective. Series in computer science. Springer, Berlin, 2005, 170 p.
859. Aggarwal C.C., Wang H. (eds) Managing and mining graph data. Advances in database systems. Springer Science – Business Media, Berlin, 2005
860. Washio T.,Motoda H. State of the Art of Graph-based Data Mining. SIGKDD Explorer Newsletter, 2003, vol. 5, no. 1, pp. 59–68
861. Yannakakis M. Graph-theoretic methods in database theory. In: Proceedings of the symposium on principles of database systems (PODS). ACM, New York, 1990, pp 230–242
862. Barcelo P., Libkin L., Reutter J. Querying graph patterns. In Proc. of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), 2011, pp. 199–210
863. Wang X. Finding patterns on protein surfaces: Algorithms and applications to protein classification. IEEE Transactions on Knowledge and Data Engineering, 2005, vol. 17, pp. 1065–1078
864. Carroll J. Matching RDF Graphs. In Proceedings of the International Semantic Web Conference (ISWC), 2002, pp. 5-15
865. Cruz I.F., Mendelzon A.O., Wood P.T. A graphical query language supporting recursion. ACM SIGMOD Record, Vol. 16, No 3, 1987, pp 323–330
866. Fan W., Li J. Ma S., Tang N., Wu Y. Adding regular expressions to graph reachability and pattern queries. iin Proc. of the IEEE 27th International Conference on Data Engineering (ICDE), 2011, pp. 39–50
867. Mendelzon A.O., Wood P.T. Finding regular simple paths in graph databases. SIAM J Comput, 1995, 24(6), pp. 1235–1258
868. Zhu A.D., Ma H., Xiao X., Luo S,. Tang Y., Zhou S. Shortest path and distance queries on road networks: towards bridging theory and practice. In: Proceedings of the international conference on man-

agement of data (SIGMOD). ACM, New York, 2013, pp. 857–868

869. Kanza Y., Sagiv Y. Flexible queries over semistructured data. PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2001, pp. 40–51.
870. Hurtado C.A., Poulovassilis A., Wood P.T. Ranking approximate answers to semantic web queries. Ranking Approximate Answers to Semantic Web Queries. In: Aroyo L. et al. (eds) The Semantic Web: Research and Applications. ESWC 2009. Lecture Notes in Computer Science, vol 5554. Springer, Berlin, Heidelberg. 2009, pp. 263–277
871. Cruz, I.F., Mendelzon, A.O., Wood, P.T. G+: Recursive Queries without Recursion. In: Proceedings of the 2th International Conference on Expert Database Systems (EDS). 1989, pp. 645–666
872. Consens, M.P., Mendelzon, A.O. GraphLog: a Visual Formalism for Real Life Recursion. In: Proceedings of the 9th ACM Symposium on Principles of Database Systems. 1990, pp. 404–416.
873. Wood, P.T.: Factoring Augmented Regular Chain Programs. In: Proceedings of the 16th International Conference on Very Large Data Bases (VLDB). 1990, pp. 255– 263. Morgan Kaufmann Publishers Inc.
874. Abiteboul S., Quass D., McHugh J., Widom J., Wiener J.L. The Lorel query language for semistructured data. International Journal on Digital Libraries, 1997, 1(1), pp. 68–88
875. Flesca, S., Greco, S.: Partially Ordered Regular Languages for Graph Queries. In: Proceedings of the 26th International Colloquium on Automata, Languages and Programming (ICALP). LNCS, 1999, pp. 321–330
876. Buneman P., M. Fernandez, Suciu D. UnQL: A Query Language and Algebra for Semistructured Data Based on Structural Recursion. The VLDB Journal, 2000, 9(1), pp. 76-110
877. Hidders A.J.H. A Graph-based Update Language for Object-Oriented Data Models. Thesis (doctoral)--Technische Universiteit Eindhoven, 2001, 217 p. - https://pure.tue.nl/ws/files/2236754/200142116.pdf
878. Cardelli L., Gardner P., Ghelli G.: A Spatial Logic for Querying Graphs. In: Proceedings of the 29th International Colloquium on Automata, Languages, and Programming (ICALP). 2002, pp. 597–610. LNCS, Springer
879. Theodoratos D. Semantic Integration and Querying of Heterogeneous Data Sources Using a Hypergraph Data Model. In: Proceedings of the 19th British National Conference on Databases (BNCOD), Advances in Databases. 2002, pp. 166– 182. LNCS, Springer
880. Leser U. A query language for biological networks. Bioinformatics, 2005, 21(2), pp. 33-39
881. Liu Y.A., Stoller S.D. Querying complex graphs. In: Proc. of the 8th Int. Symposium on Practical Aspects of Declarative Languages. 2006, pp. 16–30
882. Prud'hommeaux, E., Seaborne, A. SPARQL Query Language for RDF. W3C Recommendation. (January 15 2008)
883. Ronen R., Shmueli O. SoQL: a language for querying and creating data in social networks. In: Proceedings of the international conference on data engineering (ICDE). IEEE Computer Society, New York, 2009, pp 1595–1602
884. Dries A, Nijssen S., De Raedt L. A query language for analyzing networks. Proceedings of the 18th ACM conference on Information and knowledge, 2009, pp. 485-494
885. Rodriguez M.A. The Gremlin graph traversal machine and language (invited talk). In: DBPL 2015: Proceedings of the 15th Symposium on Database Programming Languages. ACM, New York, 2015, pp 1–10
886. San Martin M., Gutierrez C., Wood P.T. SNQL: A social networks query and transformation language. In: Barcelo, P. and Tannen, V. (eds.) Proceedings of the 5th Alberto Mendelzon International Workshop on Foundations of Data Management. CEUR Workshop Proceedings. CEUR-WS.org. 2011.
887. Cypher - Graph Query Language - http://neo4j.com/developer/cypher-query-language/

888. Barcelo P., Libkin L., Lin A.W., Wood P.T. Expressive languages for path queries over graph-structured data. ACM Transactions on Database Systems, 2012, Vol. 37, No 4, pp. 1–46
889. Santini S.: Regular Languages with Variables on Graphs. Information and Computation, 2012, Vol. 211, pp. 1–28
890. Feigenbaum L , Williams G.T., Clark K.G., Torres E. SPARQL 1.1 Protocol. W3C Recommendation. http://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/, March 21, 2013.
891. van Rest O., Hong S., Kim J., Meng X., Chafi H. PGQL: a property graph query language. In: Proceedings of the international workshop on graph data management experiences and systems (GRADES), 2013
892. Libkin L., Martens W., Vrgoc D. Querying Graph Databases with XPath. In: Proceedings of the 16th International Conference on Database Theory (ICDT), 2013, pp. 129–140
893. Brijder R., Gillis J.J.M., Van den Bussche J. (2013) The DNA query language DNAQL. In: ICDT '13: Proceedings of the 16th International Conference on Database Theory, 2013, pp. 1–9
894. Reutter J.L., Romero M., Vardi M.Y.: Regular queries on graph databases. In: Proceedings of the 18th International Conference on Database Theory (ICDT). 2015, pp. 177–194
895. GraphQL: A data query language. - https://code.fb.com/core-data/graphql-a-data-query-language/
896. Masseroli M., Pinoli P., Venco F., Kaitoua A., Jalili V., Paluzzi F., Muller H., Ceri S. GenoMetric Query Language: A novel approach to large-scale genomic data management. Bioinformatics, 2015, 31(12), pp. 1881-1888
897. Giugno R., Shasha D. GraphGrep: a fast and universal method for querying graphs. In: Proceedings of the 16th International Conference on Pattern Recognition, 2002. pp. 112–115.
898. He H., K. Singh A. Graphs-at-a-time: query language and access methods for graph databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 2008. p. 405–418.
899. Milo T., Suciu D.. Index structures for path expressions. In: Proceedings of the 7th International Conference on Database Theory; 1999. pp. 277–295
900. Picalausa F., Luo Y., Fletcher G.H.L., Hidders J, Vansummeren S. A structural approach to indexing triples. In: Proceedings of the 9th Extended Semantic Web Conference; 2012. p. 406–421
901. Trißl S., Leser U. Fast and practical indexing and querying of very large graphs. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 2007. p. 845–856.
902. Calvanese D., De Giacomo G., Lenzerini M., Vardi M.Y. Reasoning on regular path queries. SIGMOD Rec. 2003;32(4):83–92.
903. Fernandez M., Suciu D. Optimizing regular path expressions using graph schemas. In: Proceedings of the 14th International Conference on Data Engineering; 1998. p. 14–23.
904. Goldman R., Widom J. DataGuides: enabling query formulation and optimization in semistructured databases. In: Proceedings of the 23rd International Conference on Very Large Data Bases; 1997. p. 436–445.
905. Urbón P. NoSQL graph database matrix. - http://nosql.mypopescu.com/post/619181345/nosql-graph-database-matrix
906. Deepak Singh Rawat, Navneet Kumar Kashyap. Graph Database: A Complete GDBMS Survey.International Journal for Innovative Research in Science & Technology (IJIRST), 2017, Vol. 3, No 12, pp. 217-226
907. Pradeep Jadhav, Ruhi Oberoi. Comparative Analysis of Different Graph Databases, International Journal of Engineering Research & Technology (IJERT), Vol. 3, No 9, 2014, pp. 820-824
908. Stonebraker M., Madden S.R., Abadi D.J., Harizopoulos S., Hachem N.I. The End of an Architectural Era (It's Time for a Complete Rewrite). - VLDB '07: Proceedings of the 33rd international conference on Very large data basesSeptember 2007 Pages 1150–1160

909. R. Kallman, H. Kimura, J. Natkins, A. Pavlo, A. Rasin, S. Zdonik, E. Jones, S. Madden, M. Stonebraker, Y. Zhang, J. Hugg & D. Abadi, "H-store: a high-performance, distributed main memory transaction processing system," Proceedings of the VLDB Endowment, Volume 1 Issue 2, August 2008, pages 1496-1499.
910. M. Stonebraker, D. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, P. O'Neil, A. Rasin, N.Tran & S. Zdonik, "C-store: a column-oriented DBMS," Proceedings of the 31st International Conference on Very Large Data Bases (VLDB '05), 2005, pages 553 – 564.
911. Cattell Rick. "Scalable SQL and NoSQL data stores," ACM SIGMOD Record 39.4 (2011): 12-27.
912. Matthew A. (2011). "How Will The Database Incumbents Respond To NoSQL And NewSQL?". 451 Group - https://www.cs.cmu.edu/~pavlo/courses/fall2013/static/papers/aslett-newsql.pdf
913. Matthew A. (2011)." What we talk about when we talk about NewSQL". 451 Group - https://blogs.451research.com/information_management/2011/04/06/what-we-talk-about-when-we-talk-about-newsql/
914. Stonebraker Mil. NewSQL: An Alternative to NoSQL and Old SQL for New OLTP Apps. Communications of the ACM Blog. - https://cacm.acm.org/blogs/blog-cacm/109710-new-sql-an-alternative-to-nosql-and-old-sql-for-new-oltp-apps/fulltext
915. Pavlo A., Aslett M. What's Really New with NewSQL?. SIGMOD Record, June 2016, Vol. 45, No. 2. pp. 45-55
916. Venkatesh, Prasanna (January 30, 2012). "NewSQL - The New Way to Handle Big Data". - https://www.opensourceforu.com/2012/01/newsql-handle-big-data/
917. Studer R., Benjamins R., Fensel D. Knowledge engineering: Principles and methods. Data & Knowledge Engineering, 25(1–2):161–198, 1998.
918. Guarino N., Oberle D., Staab S. What is an ontology? In Handbook on ontologies, pages 1–17. Springer, 2009.
919. Alexaki S., Christophides V., Karvounarakis G., Plexousakis D., Tolle K.: The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases. In: SemWeb'01: Proceedings of the Second International Conference on Semantic Web - Volume 40 May 2001, pp. 1–13
920. Broekstra J., Kampman A., van Harmelen F. Sesame: A generic architecture for storing and querying RDFand RDF schema. In Proc. of the First Inter. Semantic Web Conf., pp. 54–68, 2002.
921. Pan Z., Heflin J.: Dldb: Extending relational databases to support semantic web queries. In: Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (PSSS'03). 2003, pp. 109–113
922. Harris S., Gibbins N. 3store: Efficient bulk RDF storage. In Proc. of the 1st Intern. Workshop on Practical and Scalable Semantic Systems (PSSS'03), 2003. pp. 1-15
923. Theoharis Y., Christophides V., Karvounarakis G. (2005) Benchmarking Database Representations of RDF/S Stores. In: Gil Y., Motta E., Benjamins V.R., Musen M.A. (eds) The Semantic Web – ISWC 2005. ISWC 2005. Lecture Notes in Computer Science, vol 3729. Springer, Berlin, Heidelberg. pp. 685-701
924. McBride B. Jena: Implementing the RDF Model and Syntax Specification. SemWeb'01: Proceedings of the Second International Conference on Semantic Web - Volume 40, May 2001, pp, 23–28
925. Agrawal R., Somani A., Xu Y. Storage and querying of e-commerce data. In: VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc. (2001) 149–158
926. Ma L., Su Z., Pan Y., Zhang M, Liu M. Rstar: an rdf storage and query system for enterprise resource management. thirteenth ACM international conference on Information and knowledge management, 2004:484 – 491.
927. Erling O., Mikhailov I.: RDF Support in the Virtuoso DBMS. In: Conference on Social Semantic Web (CSSW'07). Volume 113. (2007) 59–68
928. Wu Z., Eadon G., Das S., Chong E.I., Kolovski, V., Annamalai, M., Srinivasan,

J.: Implementing an Inference Engine for RDFS/OWL Constructs and User-Defined Rules in Oracle. In: Proceedings of the 24th International Conference on Data Engineering (ICDE'08). (2008) 1239–1248

929. Alexaki S., Christophides V., Karvounarakis G., Plexousakis D., Tolle K. On storing voluminous rdf descriptions: The case of web portal catalogs. In Proceedings of the Fourth International Workshop on the Web and Databases, WebDB 2001, Santa Barbara, California, USA, May 24-25, 2001, in conjunction with ACM PODS/SIGMOD 2001: 43-48
930. Abadi D.J., Marcus A., Madden S.R., Hollenbach K. Scalable Semantic Web Data Management Using Vertical Partitioning. In: Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB'07). (2007) 411–422
931. Jing L., Li M.,Lei Z., Jean-Sébastien B., Chen W., Yue P., Yong Y., 2007. SOR: A Practical System for Ontology Storage, Reasoning. In VLDB 2007, 33rd Very Large Data Bases Conference,pp 1402-1405.
932. Dehainsala H., Pierra G., Bellatreche L. (2007) OntoDB: An Ontology-Based Database for Data Intensive Applications. In: Kotagiri R., Krishna P.R., Mohania M., Nantajeewarawat E. (eds) Advances in Databases: Concepts, Systems and Applications. DASFAA 2007. Lecture Notes in Computer Science, vol 4443. Springer, Berlin, Heidelberg. pp 497-508
933. Park M.J., Lee J.H., Lee C.H., Lin J., Serres O., Chung C.W.: An Efficient and Scalable Management of Ontology. In: Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07). (2007) 975–980
934. Wilkinson K., Sayers C., Kuno H., Reynolds D. 2003. Efficient RDF storage and Retrieval in Jena2. Proceedings of the 1st International Workshop on Semantic Web Database (SWDB'03). pp. 131–150.
935. SWAD-Europe Deliverable 10.2: Mapping Semantic Web Data with RDBMSes. - https://www.w3.org/2001/sw/Europe/reports/scalable_rdbms_mapping_report/
936. Bailey J., Bry F., Furche T., Schaffert S. (2005) Web and Semantic Web Query Languages: A Survey. In: Eisinger N., Małuszyński J. (eds) Reasoning Web. Lecture Notes in Computer Science, vol 3564. Springer, Berlin, Heidelberg, 2005, pp. 35–133
937. Jean S., Aït-Ameur Y., Pierra G. (2006) Querying Ontology Based Database Using OntoQL (An Ontology Query Language). In: Meersman R., Tari Z. (eds) On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE. OTM 2006. Lecture Notes in Computer Science, vol 4275. Springer, Berlin, Heidelberg. pp 704-721

937a. Melichar B., Holub J., Polcar T. Text searching algorithms Volume I: Forward string matching. Czech Technical University in Prague, 224 p. - /http://www.stringology.org/athens/TextSearchingAlgorithms/tsa-lectures-1.pdf

938. Melichar B., Holub J., Polcar T. Text searching algorithms Volume II: Backward string matching. Czech Technical University in Prague, 61 p.- http://www.stringology.org/athens/TextSearchingAlgorithms/tsa-lectures-2.pdf
939. Hakak S., Kamsin A., Shivakumara P., Gilkar G., Khan W.Z., Imran M. Exact String Matching Algorithms: Survey, Issues, and Future Research Directions. IEEE Access, 2019, Vol. 7, pp 69614-69637

941. Christian Charras, Thierry Lecroq. Handbook of exact string matching algorithms. College Publications (February 27, 2004), 256 p. - http://www-igm.univ-mlv.fr/~lecroq/string/string.pdf
942. Faro S. Exact Online String Matching Bibliography. 2016, 23 p. - https://arxiv.org/abs/1605.05067
943. Faro S., Lecroq T., Borzi, Di Mauro S., Maggio A.. The String Matching Algorithms Research Tool. Stringology 2016: 99-111
944. Faro S. Lecroq T,"The exact online string matching problem: A review of the most recent results", ACM Comput. Survey, Article 13, 42 pages, February 2013.
945. Koloud Al-Khamaiseh, Shadi AL Shagarin. A Survey of String Matching Algo-

rithms. Int. Journal of Engineering Research and Applications, 2014, vol. 4, No 7, pp.144-156

946. Morris, J.H., Jr; Pratt, V. (1970). A linear pattern-matching algorithm (Technical report). University of California, Berkeley, Computation Center. TR-40.
947. Knuth, Donald E. (1973). "The Dangers of Computer-Science Theory". Studies in Logic and the Foundations of Mathematics. 74: 189–195.
948. Knuth D., Morris J.H., Pratt V. (1977). "Fast pattern matching in strings". SIAM Journal on Computing. 6 (2): 323–350.
949. Matiyasevich, Yuri (1973). "Real-time recognition of the inclusion relation". Journal of Soviet Mathematics. 1: 64–70
950. Boyer R.S., Moore J.S. A fast string searching algorithm. Communications of the ACM. 1977, vol. 20, No 10. pp. 762—772. — doi:10.1145/359842.359859.
951. Baeza-Yates R., Gonnet G.H. A new approach to text searching. Communications of the ACM, Vol. 35, No 10,1992 pp 74–82. - https://doi.org/10.1145/135239.135243
952. Алгоритм Бойера-Мура - https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%91%D0%BE%D0%B9%D0%B5%D1%80%D0%B0_%E2%80%94_%D0%9C%D1%83%D1%80%D0%B0
953. Horspool R.N. Practical fast searching in strings, Software - Practice & Experience, 1980, 10(6) :501-506.
954. Zhu R.F., Takaoka T., 1987, On improving the average case of the Boyer-Moore string matching algorithm, Journal of Information Processing 10(3):173-177.
955. Turbo-BM algorithm - http://www-igm.univ-mlv.fr/~lecroq/string/node15.html
956. CROCHEMORE, M., CZUMAJ A., GASIENIEC L., JAROMINEK S., LECROQ T., PLANDOWSKI W., RYTTER W., 1992, Deux méthodes pour accélérer l'algorithme de Boyer-Moore, in Théorie des Automates et Applications, Actes des 2e Journées Franco-Belges, D. Krob ed., Rouen, France, 1991, pp 45-63, PUR 176, Rouen, France.
957. Apostolico A., Giancarlo R. "The Boyer-Moore-Galil String Searching Strategies Revisited," (in English), SIAM Journal on Computing, vol. 15, No. 1, pp. 98-105, Feb 1986.
958. Smith P.D., "Experiments with a very fast substring search algorithm," Software-Practice and Experience, vol. 21, no. 10, pp. 1065-1074, 1991.
959. Raita T. Tuning the Boyer-Moore-Horspool string searching algorithm. Software-Practice and Experience, vol. 22, no. 10,pp. 879-884, 1992.
960. Crochemore M., Czumaj A., Gasieniec L., Jarominek S., Lecroq T., Plandowski W. Rytter W. "Speeding up two string-matching algorithms," Algorithmica 12(4-5):247-267, 1994.
961. Berry T., Ravindran, S. (2001) A Fast String Matching Algorithm and Experimental Results. Proceedings of the Prague Stringology Club Workshop '99, Collaborative Report DC-99-05, Czech Technical University, Prague, 16-26.
962. Sunday D.M. " A very fast substring search algorithm," Communications of the ACM, Vol. 33, No 8, 1990 pp 132–142. - https://doi.org/10.1145/79173.79184.
963. Colussi L. Correctness and efficiency of pattern matching algorithms. Information and Computation, vol. 95, no. 2, pp. 225-251, 1991.
964. Xian-feng H., Yu-bao Y., Xia L. "Hybrid pattern-matching algorithm based on BM-KMP algorithm." (ICACTE) 2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE), 2010, pp. V5-310-V5-313, DOI: 10.1109/ICACTE.2010.5579620.
965. Cao Z., Zhenzhen Y., Lihua L. "A fast string matching algorithm based on low-light characters in the pattern." In Advanced Computational Intelligence (ICACI), 2015 Seventh International Conference on, pp. 179-182. IEEE, 2015.
966. Hakak S., Kamsin A., Shivakumara P., Idna Idris M.Y., Gilkar G.A. "A new split based searching for exact pattern matching for natural texts." PloS ONE 13, no. 7 (2018): e0200912. Skid
967. Hakak S., Amirrudin K., Shivakumara P., Idna Idris M.Y. "Partition-Based Pattern Matching Approach for Efficient Retrieval Of Arabic Text." Malaysian

Journal of Computer Science 31, no. 3 (2018): 200-209.
968. Franek F., Jennings C.G., Smyth W.F. A simple fast hybrid pattern-matching algorithm. J. Discrete Algorithms, 5(4):682–695, 2007.
969. Rabin M.O., Karp R.M. Efficient randomized pattern-matching algorithms. IBM Journal of Research and Development. 1987, vol. 31, No 2, pp. 249–260. — doi:10.1147/rd.312.0249.
970. Rabin–Karp algorithm. - https://en.wikipedia.org/wiki/Rabin%E2%80%93Karp_algorithm
971. Wu S., Manber U. "A fast algorithm for multi-pattern searching," Department of Computer Science, University of Arizona, Tucson, AZ, Report TR-94-171994.
972. Kim S., Kim Y., "A fast multiple string pattern matching algorithm," in Proceedings of 17th AoM/IAoM Conference on Computer Science, 1999, pp. 44-49.
973. Simone F. "A very fast string matching algorithm based on condensed alphabets." In International Conference on Algorithmic Applications in Management, pp. 65-76. Springer, Cham, 2016.
974. Lecroq T. "Fast exact string matching algorithms,"Information Processing Letters, vol. 102, no. 6, pp. 229-235, Jun 15 2007.
975. Kalsi P., Peltola H., Tarhio J. "Comparison of exact string matching algorithms for biological sequences," in Proceedings of the Second International Conference on Bioinformatics Research and Development, BIRD, 2008. pp. 417-426.
976. Daciuk J., Mihov S, Watson B., Watson R. Incremental construction of minimal acyclic finite state automata. Computational Linguistics, 2000, 26(1), pp.3-16.
977. Yang. W. Mealy machines are a better model of lexical analyzers. Computer Languages, Vol. 22, No 1, 1996, pp. 27-38
978. Blumer A., Blumer J., Ehrenfeucht A., Haussler D., McConnel R. Linear size finite automata for the set of all subwords of a word: an outline of results. Bull. European Assoc. Theoret. Comput. Sci., 21:12-20, 1983
979. Commentz-Walter B. A string matching algorithm fast on the average. Proceedings of the 6th Colloquium, on Automata, Languages and Programming, 1979, pp. 118–132
980. Allauzen C., Raffinot M. Simple optimal string matching algorithm, Journal of Algorithms, Vol. 36, No 1, 2000, pp. 102–116
981. Allauzen C., Crochemore M., Raffinot M. Factor oracle: A new structure for pattern matching. In 26th Seminar on Current Trends in Theory and Practice of Informatics (SOFSEM'99), Nov 1999, Milovy, Czech Republic, Czech Republic. pp.291-306.
982. Faro S., Lecroq T. Efficient variants of the Backward-Oracle-Matching algorithm. In Proceedings of the Prague Stringology Conference, Czech Republic, 2008, pp. 146-160: Czech Technical University.
983. Fan H., Yao N., Ma H. Fast variants of the backward-oraclemarching algorithm. In Fourth International Conference on Internet Computing for Science and Engineering, 2009, pp. 56-59.
984. He L., Fang B., Sui J. The wide window string matching algorithm. Theoretical Computer Science, vol. 332, no. 1-3, 2005, pp. 391-404
985. Liu C., Wang Y., Liu D., Li D. Two improved single pattern matching algorithms. In ICAT Workshops, Hangzhou, China,, 2006, pp. 419-422: IEEE Computer Society.
986. Hongbo F., Shupeng S., Jing Z., Li D. Suffix Type String Matching Algorithms Based on Multi-windows and Integer Comparison. In International Conference on Information and Communications Security, pp. 414-420. Springer, Cham, 2015.
987. Masaki Waga, Ichiro Hasuo, Kohei Suenaga. "Efficient online timed pattern matching by automata-based skipping." In International Conference on Formal Modeling and Analysis of Timed Systems, pp. 224-243. Springer, Cham, 2017.
988. Bitap algorithm. - https://en.wikipedia.org/wiki/Bitap_algorithm
989. Bálint Dömölki, An algorithm for syntactical analysis, Computational Linguistics 3, Hungarian Academy of Science pp. 29–46, 1964.

990. Bálint Dömölki, A universal compiler system based on production rules, BIT Numerical Mathematics, 8(4), pp 262–275, 1968. doi:10.1007/BF01933436
991. Shyamasundar R.K. Precedence parsing using Dömölki's algorithm, International Journal of Computer Mathematics, 6(2) pp 105–114, 1977.
992. Baeza-Yates R., Gonnet G.H. A new approach to text searching. Communications of the ACM, Vol. 35, No 10,1992 pp 74–82
993. Ricardo A. Baeza-Yatesm Gaston H. Gonnet. A New Approach to Text Searching. Communications of the ACM, 1992, vol. 35, No 10, pp. 74-82 - ПОВТОРЕНИЕ 5a)
994. Fredriksson K., Grabowski S. Practical and optimal string matching. In SPIRE'05: Proceedings of the 12th international conference on String Processing and Information Retrieval, 2005, pp. 376–387. - https://doi.org/10.1007/11575832_42
995. Salmela L., Tarhio J., Kytojoki J. Multi pattern string matching with q-grams. Journal of Experimental Algorithms, 2006, Vol. 11, pp. 1-19
996. Udi Manber, Sun Wu. "Fast text search allowing errors." Communications of the ACM, 35(10): pp. 83–91, October 1992, doi:10.1145/135239.135244.
997. R. Baeza-Yates and G. Navarro. A faster algorithm for approximate string matching. In Dan Hirchsberg and Gene Myers, editors, Combinatorial Pattern Matching (CPM'96), LNCS 1075, pages 1–23, Irvine, CA, June 1996.
998. G. Myers. "A fast bit-vector algorithm for approximate string matching based on dynamic programming." Journal of the ACM 46 (3), May 1999, 395–415.
999. Navarro G., Raffinot M. A Bit-parallel Approach to Suffix Automata: Fast Extended String Matching. In Proc CPM'98, Lecture Notes in Computer Science 1448: 14-33, 1998.
1000. Navarro G., Raffinot M. Fast and flexible string matching by combining bit-parallelism and suffix automata. ACM Journal. Experimental Algorithmics,2000, 5(4): 1-36
1001. Peltola H., Tarhio J. Alternative Algorithms for Bit-Parallel String Matching. In String Processing and Information Retrieval, Spire Springer, LNCS 2857, pp. 80-93, 2003.
1002. Branislav Durian, Jan Holub, Hannu Peltola and Jarma Tarhio,"Tuning BNDM with q-grams", In the proc. Of workshop on algorithm engineering and experiments, SIAM USA, pp. 29-37, 2009.
1003. Miao C., Chang G., Wang X. Filtering Based Multiple String Matching Algorithm Combining q-Grams and BNDM. In ICGEC '10: Proceedings of the 2010 Fourth International Conference on Genetic and Evolutionary Computing, 2010, pp. 82–585. - https://doi.org/10.1109/ICGEC.2010.149
1004. Faro S., Lecroq T. Efficient variants of the backward-oracle-matching algorithm. International Journal of Foundations of Computer Science, vol. 20, no. 6, pp. 967–984, Dec. 2009,
1005. Peltola H., Tarhio J. (2003) Alternative Algorithms for Bit-Parallel String Matching. In: Nascimento M.A., de Moura E.S., Oliveira A.L. (eds) String Processing and Information Retrieval. SPIRE 2003. pp. 80-93. Lecture Notes in Computer Science, vol 2857. Springer, Berlin, Heidelberg.
1006. M. Oguzhan Külekci, "Filter based fast matching of long patterns by using SIMD instructions," in Proceedings of the Prague Stringology Conference, Prague, Czech Republic, 2009. pp. 118--128
1007. M. Oguzhan Külekci, "A method to overcome computer word size limitation in bit-parallel pattern matching," in Proceedings of the 19th International Symposium on Algorithms and Computation, ISAAC, 2008. pp. 496--506
1008. Gupta S., Rasool A. Bit Parallel String Matching Algorithms: A Survey. International Journal of Computer Applications, 2014, vol. 95, No 10, pp. 27-32
1009. M. Crochemore, A. Czumaj, L. Gaĺ˘gsieniec, T. Lecroq, W. Plandowski, and W. Rytter, "Fast practical multi-pattern matching," Information Processing Letters, vol. 71, no. 3-4, pp. 107-113, Aug 27 1999.
1010. G. Navarro, Nrgrep: A fast and flexible pattern matching tool. Software—Prac-

tice & Experience, Vol. 31, No 13, 2001, pp. 1265–1312. - https://doi.org/10.1002/spe.411.

1011. F. Franek, Jennings, C. G., and Smyth, W. F., "A simple fast hybrid pattern-matching algorithm," J. Discret. Algorithms, pp. 682-695, 2007.
1012. S. Deusdado and P. Carvalho, "GRASPm: an efficient algorithm for exact pattern-matching in genomic sequences," Int J Bioinform Res Appl, vol. 5, no. 4, pp. 385-401, 2009.
1013. P. Shivendra Kumar, H. K. Tiwari, and P. Tripathi. "Hybrid approach to reduce time complexity of string matching algorithm using hashing with chaining." In Proceedings of International Conference on ICT for Sustainable Development, pp. 185-193. Springer, Singapore, 2016.
1014. Hamming R. W. "Error detecting and error correcting codes". The Bell System Technical Journal. 1950, 29 (2): 147–160
1015. Levenshtein V.I. Binary codes with correction of dropouts, insertions and substitutions of symbols (RUS). Reports of the Academy of Sciences of the USSR, 1965. 163.4: 845-848.
1016. Levenshtein, Vladimir I. (February 1966). "Binary codes capable of correcting deletions, insertions, and reversals". Soviet Physics Doklady, 1966. 10 (8): 707–710.
1017. Dan Gusfield. Algorithms on stings, trees, and sequences: Computer science and computational biology ACM SIGACT News, Vol. 28, No 4, Dec. 1997, pp. 41–60. - https://doi.org/10.1145/270563.571472
1018. Damerau F.J. A technique for computer detection and correction of spelling errors. Communications of the ACM, 1964, vol. 7, No 3, pp 171–176
1019. Winkler, W. E. (1990). "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage" (PDF). Proceedings of the Section on Survey Research Methods. American Statistical Association: 354–359.
1020. Jaro, M. A. Advances in record linkage methodology as applied to the 1985 census of Tampa Florida Journal of the American Statistical Association. 1989, Vol. 84, No. 406, pp. 414-420
1021. Longest common subsequence problem. - https://en.wikipedia.org/wiki/Longest_common_subsequence_problem
1022. Hall P., Dowling G. Approximate string matching. ACM Computing Surveys, 12(4) :381-402, 1980.
1023. Sankoff D., Kruskal J., editors. Time Warps. String Edits, arid Macro molecules: The Theory arid Practice of Sequence Comparison. Add is on-Wesley, 1983.
1024. Apostolico A., Galil Z. Combinatorial Algorithms on Words. NATO ISI Series. Springer-Verlag, 1985.
1025. Galil Z., Giancarlo R. Data structures and algorithms for approximate string matching. Journal of Complexity, Vol. 4, No 1, 1988, pp. 33-72
1026. Jokinen P, Tarhio J, Ukkonen E. A comparison of approximate string matching algorithms. Software Practice arid Experience, 26(12): 1439-1458,1996.
1027. Navarro G. A guided tour to approximate string matching. ACM Computing Surveys, Vol. 33, No , 1, 2001, pp 31–88
1028. Syeda Shabnam Hasan, F. Ahmed, Rosina Surovi Khan. Approximate String Matching Algorithms: A Brief Survey and Comparison. International Journal of Computer Applications, 2015, Vol. 120, No. 8, pp. 26-31
1029. Licklider J.C.R. Libraries of the future. Cambridge, MA: The MIT Press; 1965.
1030. Charles P. Bourne, Trudi Bellardo Hahn. A History of Online Information Services, 1963-1976. MIT Press, 2003, 496 p
1031. Project Gutenberg. - https://en.wikipedia.org/wiki/Project_Gutenberg
1032. Schatz B. (1996). Chen H. (ed.). "Building large-scale digital libraries". IEEE Computer. 29 (5): 22–25.
1033. Functional Requirements for Bibliographic Records, Final Report / IFLA Study Group on the Functional Requirements for Bibliographic Records. – München: K.G. Saur, 1998.
1034. Crofts N., Doerr M., Gill T., Stead S.,, Stiff M. (editors), Definition of the CIDOC Conceptual Reference Model, January 2008. Version 4.2.4.
1035. CERIF in Brief. - https://www.eurocris.org/eurocris_archive/cerifsupport.org/cerif-in-brief/index.html

1036. David Shotton. Introduction the Semantic Publish-ing and Referencing (SPAR) Ontologies. October 14, 2010. http://opencitations.wordpress.com/ 2010/10/14/introducing-the-semantic-publishing-and-referencing-spar-ontologies/
1037. Candela L., Castelli D., Fuhr N., Ioannidis Y., Klas C.-P., Pagano P., Ross S., Saidis C., Schek H.-J., Schuldt H., Springmann M. Current Digital Library Systems: User Requirements vs Provided Functionality. IST-2002- 2.3.1.12. Technology-enhanced Learning and Access to Cultural Heritage. March 2006.
1038. Candela L., Castelli D., Ioannidis Y., Koutrika G., Pagano P., Ross S., Schek H.J., Schuldt H. Setting the foundations of digital libraries: the DELOS manifesto. D-Lib Mag. 2007;13(3/4)
1039. Candela L., Castelli D., Dobreva M., Ferro N., Ioannidis Y., Katifori H., Koutrika G., Meghini C., Pagano P., Ross S., Agosti M., Schuldt H., Soergel D. The DELOS Digital Library Reference Model Foundations for Digital Libraries. IST-2002-2.3.1.12. Technology-enhanced Learning and Access to Cultural Heritage. Version 0.98, December 2007.
1040. Goncalves M.A., Fox E.A.., Watson L.T. and Kipp N.A. Streams, structures, spaces, scenarios, societies (5S): A formal model for digital libraries. ACM Transactions on Information Systems. 22(2), 2004, p. 270–312.
1041. Isah A., Serema B. C., Mutshewa A., Kenosi L. (2013). "Digital Libraries: Analysis of Delos Reference Model and 5S Theory". Journal of Information Science Theory and Practice. 1 (4): 38–47
1042. Open Archives Initiative Protocol for Metadata Harvesting. - https://en.wikipedia.org/wiki/Open_ArchivesInitiative_Protocol_for_Metadata_Harvesting

Отримано: 17.06.2022

Про автора:

Резніченко Валерий Анатолієвич,
кандидат фізико-математичних наук,
заступник завідувача відділом.
Кількість публікацій
в українських виданнях– 61.
Кількість зарубіжнихих публікацій – 4.
Індекс Хірша – 12.
http://orcid.org/0000-0002 4451-8931.

Місце роботи автора:

Інститут програмних систем
НАН України, 03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559.
E-mail: reznich@isofts.kiev.ua