НАЦІОНАЛЬНА
АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ПРОБЛЕМИ ПРОГРАМУВАННЯ

науковий журнал

№ 1 ***січень - березень*** ***2022***

Заснований у березні 1999 р.

ЗМІСТ

Свідоцтво про державну реєстрацію КВ № 7490 від 01.07.2003

Науковий журнал “Проблеми програмування” занесений до переліку наукових фахових видань України, в яких можуть публі-куватися основні результати дисертаційних робіт.

ISSN 1727-4907

© Інститут програмних систем НАН України 2022

NATIONAL ACADEMY
OF SCIENCES OF UKRAINE
INSTITUTE OF SOFTWARE SYSTEMS

PROBLEMS OF PROGRAMMING

scientific journal

№ 1 ***January – March*** ***2022***

Founded in March, 1999

CONTENTS

УДК 004.94; 004.4; 004.62 https://doi.org/10.15407/pp2022.01.003

Е.В. Жаріков, С.Ф. Теленик

СПЕЦІАЛІЗОВАНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОДЕЛЮВАННЯ ДИНАМІЧНОЇ КОНСОЛІДАЦІЇ ВІРТУАЛЬНИХ МАШИН

Для багатьох провайдерів хмарних послуг віртуальні машини залишаються базовою технологією віртуалізації обчислень. Віртуальні машини використовуються як для розміщення прикладних програмних засобів, так і для реалізації контейнерної віртуалізації. В умовах широкого використання віртуальних машин виникає необхідність розроблення спеціалізованого програмного забезпечення, що дозволяє визначати вплив параметрів моделей і методів управління на показники якості процесу консолідації, що дозволить запобігти виконанню експериментальних досліджень у виробничих умовах з метою оцінки нових стратегій управління ресурсами хмарного центру оброблення даних (ЦОД). Останніми роками в літературі запропоновано різні набори програмних інструментів і фреймворків для моделювання роботи ЦОД, забезпечуючи платформу та необхідні будівельні блоки для оптимізації процесу консолідації віртуальних машин. Моделі та програмні засоби моделювання процесів управління ресурсами ЦОД зазвичай не є вичерпними і вирішують конкретну проблему або завдання управління. Запропоноване у статті спеціалізоване програмне забезпечення моделювання дозволяє дослідити різні режими управління динамічною консолідацією віртуальних машин, забезпечує протоколювання результуючої інформації, такої, як показники продуктивності та діаграми навантажень, а також дозволяє визначати оптимальні параметри моделі для різних режимів роботи ЦОД, мінімізуючи кількість активних фізичних серверів та зменшуючи кількість порушень SLA.

Ключові слова: консолідація віртуальних машин, віртуалізація, хмарні обчислення, діаграма класів, діаграма послідовності.

Вступ

Інструменти моделювання хмарних центрів оброблення даних (ЦОД) широко застосовуються для досліджень і практичного використання постачальниками хмарних послуг, щоб відтворити поведінку певної групи фізичних серверів (ФС) та віртуальних машин (ВМ) у ЦОД із застосуванням нових моделей, політик, та з метою дослідження нових алгоритмів і методів управління.

Такі програмні засоби моделювання протягом багатьох років широко використовувалися в різних дослідженнях у сфері управління ресурсами хмарних ЦОД, оскільки вивчення нових підходів із використанням виробничого середовища є ризикованим і дорогим процесом, який може істотно заважати нормальній роботі хмарних сервісів у виробничих умовах. Програмне забезпечення, яке реалізує подання системи у вигляді симуляційної моделі і поводить себе аналогічно або працює з деякими припущеннями, [1] будемо називати інструментом моделювання.

Запропонований у статті інструмент моделювання ЦОД використовує модель системи, політики управління, вхідне робоче навантаження з реального ЦОД та кількість ресурсів ФС в ЦОД. Використовуючи цю інформацію, інструмент моделювання ЦОД дозволяє моделювати процеси створення, видалення та міграції ВМ, а також отримувати такі показники якості: кількість порушень угоди про рівень обслуговування (SLA, Service Level Agreement), кількість активних ФС та кількість міграцій ВМ у процесі моделювання під час експерименту.

Іншим важливим фактором, який слід мати на увазі при моделюванні хмарного ЦОД, є кількість вхідних параметрів, які враховуються під час оцінки роботи методів управління ресурсами. Багато фреймворків і програмних засобів моделювання враховують лише робоче навантаження центрального процесора (ЦП) як вхідні дані для змодельованої ВМ. Розроблений інструмент моделювання ЦОД використовує як вхідні

© Е.В. Жаріков, С.Ф. Теленик, 2022

дані для кожної змодельованої ВМ навантаження на чотири ресурси: ЦП, оперативну пам'ять, мережу та сховище.

Для розв'язання задачі моделювання роботи хмарного ЦОД з метою визначення близьких до оптимальних параметрів управління у статті пропонується спеціалізоване програмне забезпечення (ПЗ) моделювання ЦОД, що дозволяє досліджувати залежність показників якості управління ресурсами ЦОД відповідно до налаштувань змінних методу управління та параметрів моделі.

1. Аналіз публікацій

Згідно з оглядом літератури, представленим у [2, 3], інструменти та фреймворки моделювання роботи хмарного ЦОД характеризуються різними показниками, точністю моделі та архітектурною гнучкістю. Аналіз різних фреймворків моделювання показує, що надані функції і можливості недостатньо описані для розробників, які вносять модифікації з метою покращення моделі та методів управління. Крім того, при порівнянні інструментів і фреймворків моделювання слід враховувати кілька показників: функціональність, вичерпність моделі, масштабованість, вхідні дані, гнучкість, точність моделі.

CloudSim [4] – це широко використовуваний модульний і розширюваний симулятор із відкритим кодом, призначений для моделювання будь-якої можливої функціональності хмарного ЦОД та реалізації алгоритмів і методів управління для різних хмарних середовищ. Багато різних фреймворків моделювання хмарних ЦОД, запропонованих у літературі, засновані на CloudSim.

Автори [5] запропонували розширення CloudSim, щоб розробники могли внести модифікації та випробувати певну стратегію управління ресурсами з різними характеристиками та показниками якості рівня обслуговування, використовуючи георозподілені хмарні середовища. Розподілена система моделювання CloudSimScale, яка запропонована у [6], розроблена на базі CloudSim та забезпечує масштабованість системи у розподіленому середовищі, взаємодію між модулями та дозволяє моделювати процеси управління ресурсами хмарних ЦОД з використанням алгоритмів користувача. GPUCloudSim [7] – це ще одне розширення CloudSim, розроблене для аналізу та дослідження роботи політик управління з надання віртуальних машин із підтримкою GPU. Цей симулятор дозволяє моделювати розподіл ресурсів на рівні графічного процесора, щоб дослідити взаємодію між запущеними програмами користувача, накладні витрати на віртуалізацію та енергоспоживання графічних процесорів.

У [8] автори застосували моделювання дискретних подій для реалізації моделі хмарної системи, щоб дослідити продуктивність хмарної системи та отримати комбінований результат планування ВМ та відмов компонентів. Запропонований симулятор дискретних подій використовується для планування виконання програмних модулів забезпечення наукових процесів у хмарних системах та використовує продуктивність системи як цільову функцію.

Система моделювання інфраструктури як сервісу (IaaS) DISSECT-CF була запропонована у [9] і реалізує такі функції: підтримка розширюваності, підтримка оцінки споживання енергії при наданні IaaS і можливість оцінки багатьох параметрів планування та політик управління, пов'язаних із IaaS.

У [10] запропоновано систему моделювання з відкритим вихідним кодом CloudSim Plus. Автори обґрунтовують такі її переваги, як розширюваність та можливість повторного використання компонентів пропонованого фреймворку. Система моделювання забезпечує розширюваний, модульний та точний інструмент для оцінки алгоритмів управління ресурсами хмарних ЦОД.

2. Постановка задачі

Дослідження нових моделей і методів управління ресурсами хмарних ЦОД можна здійснити у спосіб розроблення спеціалізованих програмних засобів моделювання, які підтримують моделювання віртуалізованих систем, візуалізацію, реєстрацію показників якості управління та оцінку запропонованих алгоритмів управління на

змодельованій великомасштабній хмарній інфраструктурі.

Отже, необхідно розробити інструмент моделювання ЦОД, що забезпечує різні режими моделювання на основі гібридного підходу консолідації віртуальних машин, який об'єднує метод управління розміщенням нових віртуальних машин та метод управління міграцією активних віртуальних машин, мінімізуючи кількість активних фізичних серверів та зменшуючи кількість порушень SLA.

3. Розроблення моделі ЦОД з урахуванням динаміки створення, вимкнення та міграції віртуальних машин

У статті використовується модель системи, представлена в попередній статті авторів [11]. ЦОД складається з M ФС та N ВМ, $N, M \in \mathbb{N}$. Кожен ФС оснащений постійною кількістю ресурсів k таких, як ЦП, оперативна пам'ять, мережа та сховище, $k \in \{CPU, RAM, NET, IO\}$ [11].

Змінні моделі визначаються так: c_j^k – необхідна ємність ресурсу k для j-ї ВМ, C_i^k – ємність ресурсу k i-го ФС визначена складом обладнання, u_i^k – навантаження на ресурс k i-го ФС, v_i – кількість ВМ, розміщених на i-му ФС, $w^k \in [0,1]$ – відносна вага ресурсу k, $r_{\max}^k$ – найбільша необхідна ємність кожного ресурсу k серед усіх ВМ для нормалізації, $r_j = \sum_k \left(w^k c_j^k / r_{\max}^k \right)$ – необхідна загальна ємність ресурсів для j-ї ВМ, $R_{\max}^k$ – найбільша наявна ємність кожного ресурсу k серед усіх ФС для нормалізації, $R_i = \sum_k \left(w^k C_i^k / R_{\max}^k \right)$ – наявна ємність ресурсів i-го ФС, $u_i = \sum_k \left(w^k u_i^k / R_{\max}^k \right)$ – завантаженість ресурсів i-го ФС, $T^k \in [0,1]$ – порогове значення доступного ресурсу k i-го ФС, $D^k \in [0,1]$ – бажане навантаження на ресурс k, $L^k \in [0,1]$ – бажане навантаження на ресурс k i-го ФС, який визначений як недовантажений, $Q^k \in (T^k, 1]$ – доступне порогове значення ресурсу k i-го ФС, який може приймати мігруючі ВМ, v_i – кількість ВМ, що працюють на i-му ФС, $d_i = \sum_k \left(w^k d_i^k / R_{\max}^k \right)$ – відхилення від бажаного навантаження i-го ФС, $d_i^k = | c_j^k + u_i^k - C_i^k D^k |$ – відхилення від бажаного навантаження для ресурсу k, $f^k = (1 - T_k) \sum_{i \in B} (C_i^k - u_i^k) - c_m^k \sum_{i \in A} (v_i)$ – загальний обсяг ресурсу, який доступний для виконання міграцій ВМ, c_m^k – ємність ресурсу k, який необхідний для забезпечення міграції ВМ, $\theta = \sum_k \left(w^k \theta^k / \theta_{\max}^k \right)$ – оцінка можливості виконання міграції ВМ, де

$$\theta^k = \begin{cases} u_i^k < L^k C_i^k : L^k C_i^k - u_i^k \to \max \\ u_i^k \geq L^k C_i^k : -\left| D^k C_i^k - u_i^k \right| \to \max \end{cases},$$

V – список нових ВМ для розміщення в системі, W – список працюючих ФС, W^f – список ФС зі списку W, які мають вільні ресурси, R – список ФС, які знаходяться в режимі енергозбереження (очікування, сну) $|W| + |R| = M$, $W \cap R = \varnothing$.

У статті авторів [11] запропоновано два методи, що забезпечують розміщення нових ВМ та міграцію працюючих ВМ. Перший запропонований метод забезпечує ефективне розміщення нових ВМ, а другий метод консолідує існуючі ВМ, застосовуючи міграції. Рис. 1 ілюструє алгоритм для першого методу початкового розміщення ВМ. Міграція ВМ використовується або для перемикання ФС у стан енергозбереження (або стан очікування), або для розвантаження перевантажених ФС.

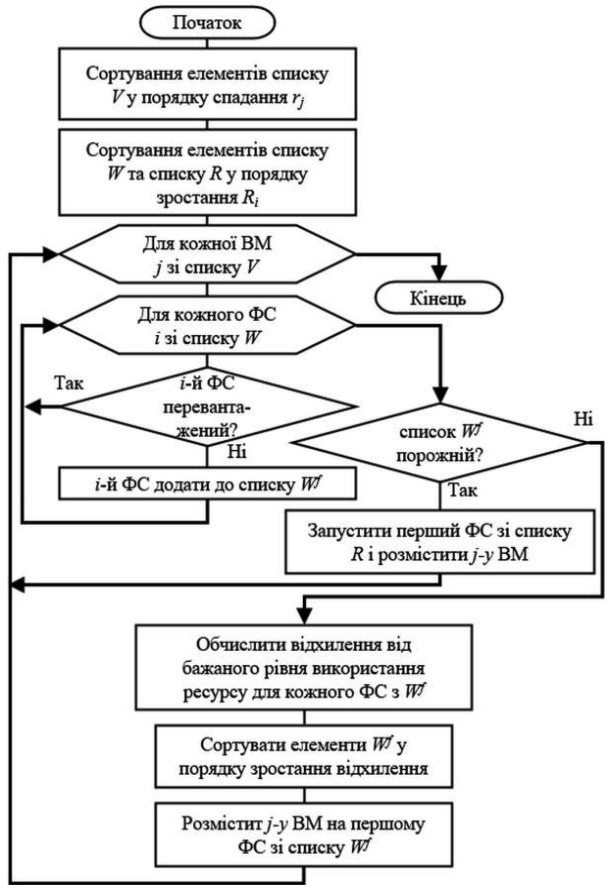


Рис. 1. Схема алгоритму розміщення нових ВМ

Міграція ВМ відбувається через перевантаження ФС (вільний обсяг ресурсу k менше ніж T^k) для одного або кількох ресурсів. Перевантаження ФС призводить до нестачі ресурсів для забезпечення роботи віртуальних машин, що призводить до порушення SLA. Для кожного ФС з набору A запропонований метод шукає такий ФС з набору B, який може приймати мігруючі віртуальні машини. Тоді один або кілька ФС з набору A можна перевести в режим енергозбереження. У результаті *першого етапу методу* отримують список A та список B [11] для подальшого *визначення плану міграції на другому етапі*. Рис. 2 ілюструє алгоритм для другого методу, спрямованого на виконання консолідації віртуальних машин.

4. Архітектура програмного забезпечення моделювання

Розроблений інструмент моделювання реалізований у вигляді ПЗ, яке забезпечує різні варіанти моделювання та дозволяє визначити близькі до оптимальних параметри моделі для різних режимів роботи ЦОД. ПЗ використовує багатошарову архітектуру та підхід до проєктування, орієнтованого на домен (domain-driven design). Архітектура складається з: (i) рівня бази даних, що забезпечує інтерфейси для отримання даних з бази даних, (ii) рівня основної логіки та (iii) рівня подання, що забезпечує інтерфейси для виводу результатів моделювання у консоль, записування результатів у текстові файли та створення файлів MS Excel з результатами моделювання.

Основним шаблоном розроблення застосунку є шаблон проєктування фасаду [12]. Він приховує складність системи та надає клієнтові інтерфейс, за допомогою якого той може отримати доступ до системи [12]. Клас фасаду відповідає за отримання даних із рівня бази даних, керує чотирма модулями та передає результати на рівень подання.

Шаблон репозиторію [13] використовується для взаємодії з рівнем бази даних, застосовуючи модель домену, щоб спростити складну бізнес-логіку та відокремити бізнес-логіку від даних. Його також можна використовувати для доступу до джерела даних з багатьох місць і для застосування централізовано керованих правил і логіки доступу [13]. Репозиторій є посередником між рівнем джерела даних і бізнес-рівнями програмного забезпечення. Репозиторій відокремлює бізнес-логіку від взаємодії з основним джерелом даних [13].

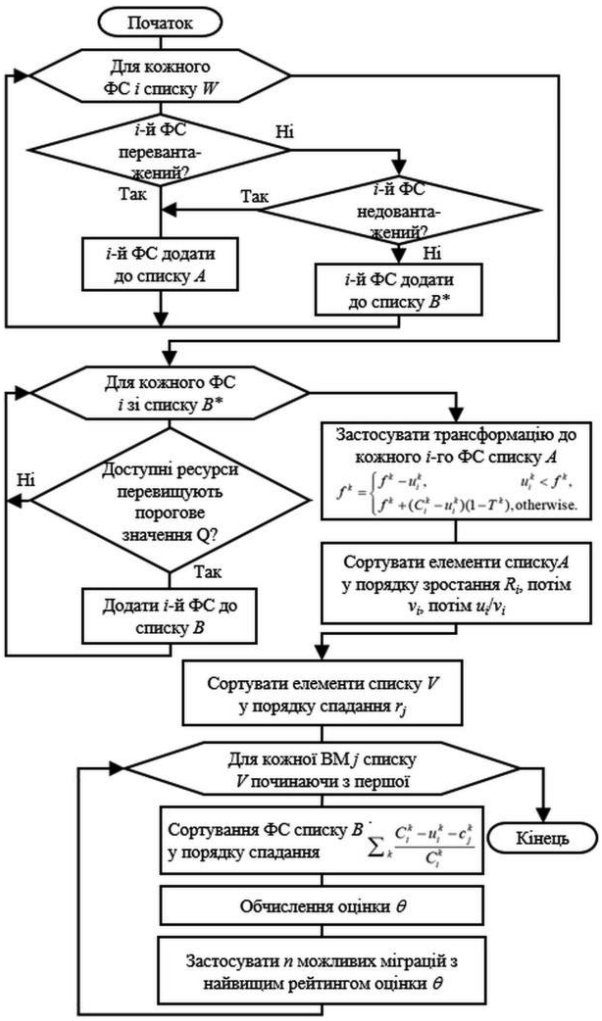


Рис. 2. Схема алгоритму консолідації ВМ

4.1. Діаграма класів

На рис. 3 показана діаграма основних класів розробленого ПЗ моделювання. Клас Server складається з основних методів, які обробляють віртуальні машини, що знаходяться у колекції VMCollection та управляють їхнім станом. Класи колекції містять ВМ та ФС, які обслуговують запити під час запуску симуляції.

Клас Simulation, показаний на рис. 4, керує чотирма модулями, а саме діагностичним модулем, який виявляє перевантажені

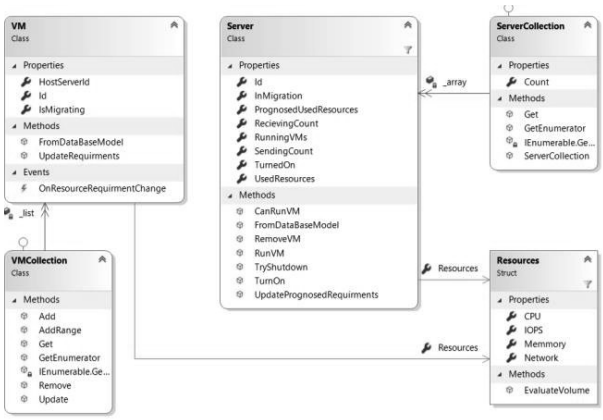


Рис. 3. Діаграма класів ПЗ моделювання

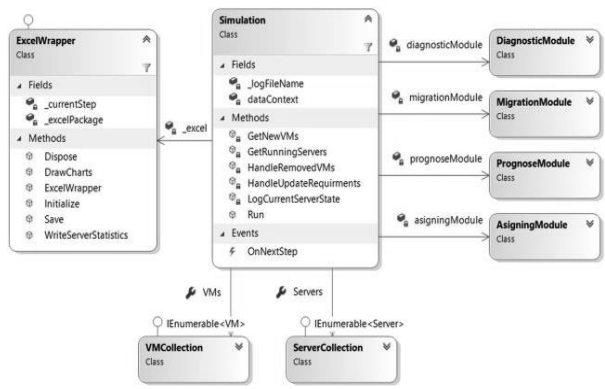


Рис. 4. Представлення класу фасаду

та недовантажені ФС, модулем розміщення нових віртуальних машин, модулем міграції, який консолідує існуючі віртуальні машини, застосовуючи міграції, і модулем прогнозування.

4.2. Діаграма послідовності

Діаграма послідовності на рис. 5 ілюструє взаємодію між основними модулями ПЗ моделювання. На першому кроці обробляються системні події, а саме: завершення роботи ВМ, розміщення нової ВМ, оновлення вимог до ресурсів для існуючих ВМ. Потім обчислюються прогнози використання ресурсів ФС для наступних кроків моделювання (один або два кроки) за допомогою модуля прогнозування. На наступному етапі працює діагностичний модуль, визначаючи перевант-

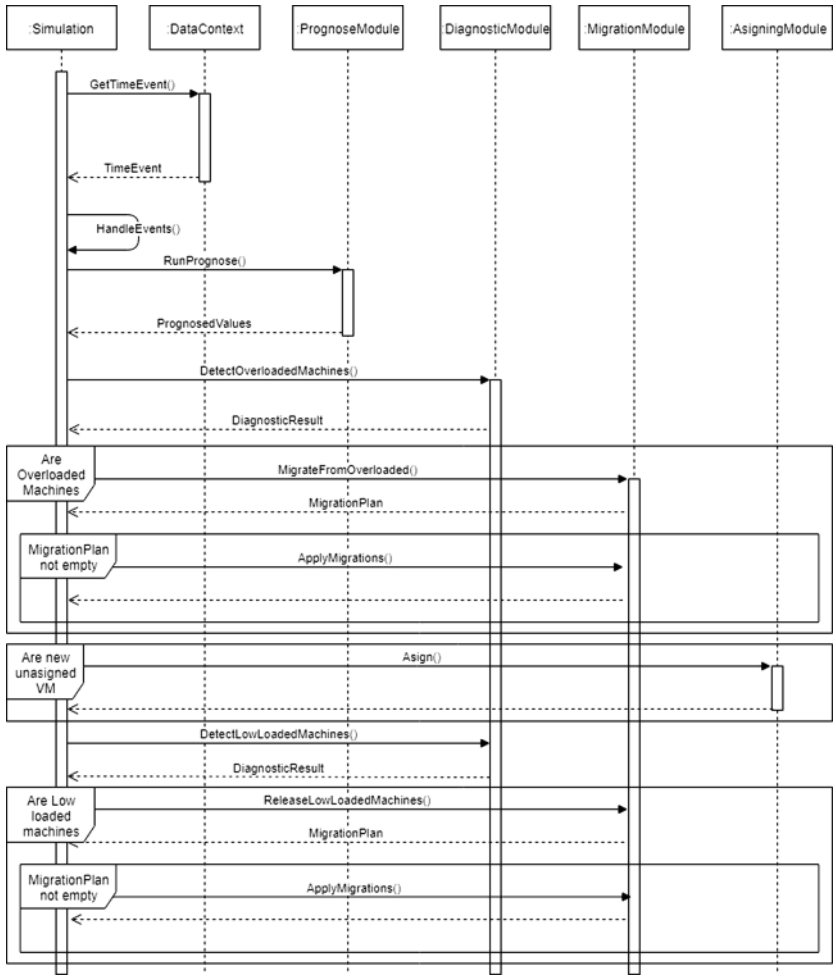


Рис. 5. Діаграма послідовності ПЗ моделювання

тажені та недовантажені ФС, які будуть передані модулю міграції ВМ для подальшого розвантаження ФС.

Якщо потрібно розмістити нові віртуальні машини, цільовий модуль виділяє багатовимірні ресурси ФС. Модуль міграції розвантажує недовантажені та перевантажені ФС. Після цього моделювання переходить до наступного кроку.

4.3. Пакети у складі ПЗ моделювання

ПЗ моделювання динамічної консолідації віртуальних машин складається з трьох пакетів, як показано на рис. 6: DAL, Utilities та Simulation. Пакет Utilities містить допоміжні класи, які використовуються в пакетах DAL і Simulation.

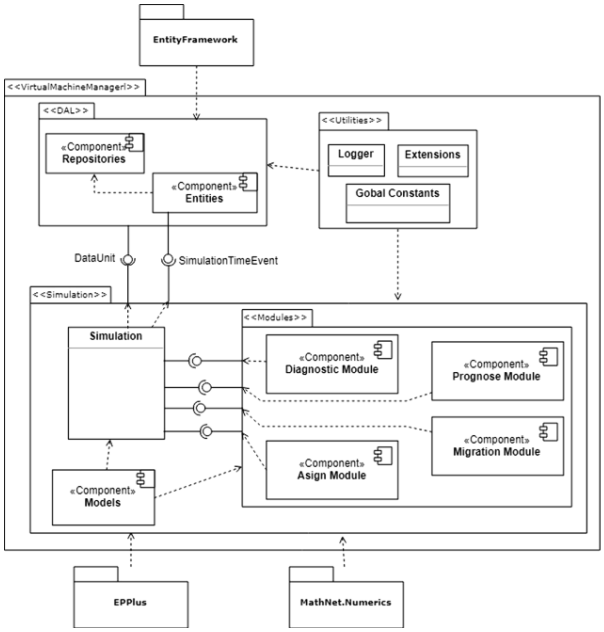


Рис. 6. Діаграма компонентів ПЗ моделювання

Пакет DAL містить логіку взаємодії з базою даних і реалізує пакет Simulation з інтерфейсами DataUnit (містить набір репозиторіїв із відповідного компонента) і SimulationTimeEvent (основна сутність, що описує всі події на певному кроці моделювання).

Пакет Simulation містить компонент із моделями ВМ і ФС, якими керує система, і пакет із чотирма модулями, що надають інтерфейси для моделювання головного класу фасаду. Крім того, пакет DAL використовує додатковий компонент EntityFramework, а пакет Simulation використовує EPPlus і MathNet.Numerics відповідно.

5. Моделювання динамічної консолідації віртуальних машин

ПЗ моделювання для динамічної консолідації ВМ реалізовано за допомогою C#, .NET Framework 4.6 [14], SQL Server Express, ORM Framework, Entity Framework, Math.NET Numerics та EPPlus для створення звітів у файлах MS Excel за допомогою .NET [15].

Для моделювання та дослідження різних режимів роботи ЦОД на основі гібридного підходу консолідації віртуальних машини автори використовують вхідні дані Bitbrains [16]. Дані файлів Bitbrains містять записи у вигляді часових рядів про навантаження на чотири типи ресурсів *k*. Робоче навантаження, записане у файли Bitbrains, являє собою використання ресурсів веб-серверів, серверів баз даних або серверів додатків 1250 віртуальними машинами, які належать до 44 різних класів [17].

У процесі моделювання автори використовують 20 гетерогенних ФС із параметрами, наведеними в таблиці 1 [18]. Крім

Табл. 1. Конфігурації ФС

PM	Number	CPUMHz	The number of PEs	RAM, GB	Storage performance, IOPS	Net, Gb/s
PowerEdge R940	5	2500	112	384	32000	40
PowerEdge R740	5	2500	56	192	23190	40
PowerEdge R830	5	2200	88	256	8000	40
PowerEdge R630	5	2200	44	128	6000	40

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Step		CPU			Memory			Network			IOPS		VM Count	Migrations	
2		|+0|	|+1|	|+2|	|+0|	|+1|	|+2|	|+0|	|+1|	|+2|	|+0|	|+1|	|+2|		Send	Recieve
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	2	700	0	0	0	0	0	7000000	0	0	0	0	0	0	0	7
5	3	631,20	0	0	336239,72	0	0	6000005,50	0	0	3,73	0	0	2	0	6
6	4	7768,40	0	0	6813637	0	0	2000637,13	0	0	2782,33	0	0	6	0	2
7	5	8983,46	0	0	7905547	0	0	1000710,31	0	0	4018,07	0	0	7	0	1
8	6	10420,07	12086,28	14018,93	8459895,00	8459895,00	8459895,00	1000647,88	1000647,88	1000647,88	3678,27	2241,86	425,98	7	0	1
9	7	16732,26	22354,99	30793,61	10701749,00	10701749,00	10701749,00	869,21	869,21	249,00	3362,73	2792,97	2170,56	8	0	0
10	8	9813,06	13990,42	12886,04	33325128,00	33325128,00	33325128,00	8000598,00	229,65	117,53	2384,67	1461,28	719,50	8	8	0
11	9	9012,40	8688,31	8428,44	12090769,00	12090769,00	12090769,00	4000425,50	4000425,50	4000425,50	3224,27	3306,69	3166,44	4	4	0
12	10	259,80	789,19	625,44	2172644,50	2147479,00	2147479,00	1000004,50	1000004,31	1000004,31	75,53	-516,50	-525,45	2	1	0
13	11	281,80	223,20	189,34	581607,13	581607,13	581607,13	1000004,63	1000004,63	1000004,63	70,47	55,01	45,36	1	1	0
14	12	967,60	623,80	525,54	760563,88	760563,88	760563,88	1000008,88	1000008,88	1000008,88	56,93	42,74	33,36	1	1	0
15	13	402,40	326,82	192,42	1374152,88	1342173,50	1342173,50	1000004,50	1000004,50	1000004,50	31,53	25,68	16,41	4	1	0

Рис. 7. Навантаження на ресурси одного ФС

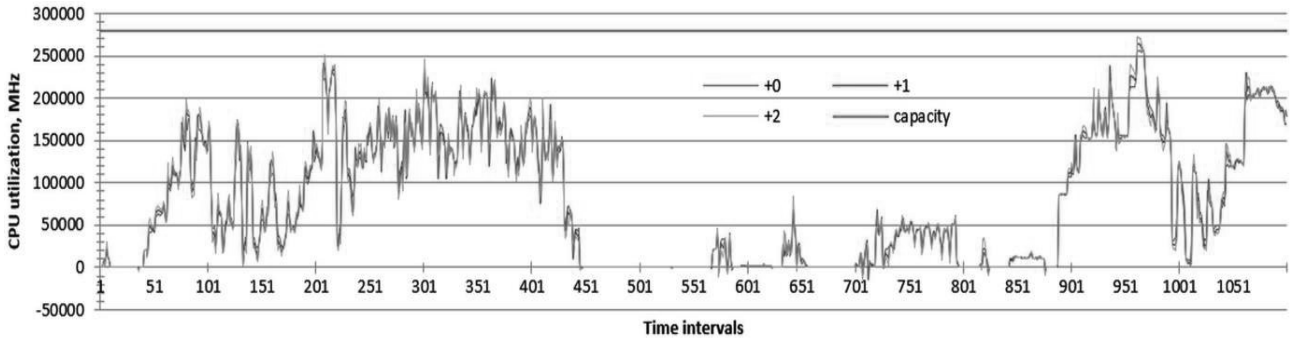


Рис. 8. Використання ЦП ФС 1 (Power-Edge R940) у файлі MS Excel

того, автори використовують дані Bitbrains, зібрані з 1114 віртуальних машин із різним обсягом запитуваного ЦП (у МГц), пам'яті (у МБ), сховища (в IOPS) та мережі (у Гбіт/с), які подаються у застосунок моделювання протягом 1100 часових інтервалів.

Параметри моделі ЦОД для кожної симуляції, показані в таблиці 2, зберігаються у файлі settings.ini.

Табл. 2. Параметри моделі

Parameter	Variable	Value
TIME_STEP_VALUE	-	300, s
STEPS_TO_SIMULATE	-	1100
RES_WEIGHT	w^k	1
RES_THREADHOLD	T^k	Changes
RES_DESIRED_LEVEL	D^k	Changes
RES_LOW_LEVEL	L^k	Changes
RES_RECIEVER_ THREADHOLD	Q^k	Changes
BEAM_LENTH	N	Changes
CPU_ON_MIGRATION		100, MHz
MIN_NETWORK_ON_ MIGRATION		100, Mbps

Розроблене ПЗ моделювання генерує файл MS Excel, в якому зберігається використання ресурсів, кількість розміщених ВМ і кількість міграцій ВМ для кожного ФС під час виконання симуляції, як показано на рис. 7. Файл MS Excel також містить діаграми використання усіх ресурсів під час симуляції. На рис. 8 показані фактичні, прогнозовані значення та загальна продуктивність ЦП фізичного сервера Power-Edge R940.

У застосунку використовується регресійна модель для прогнозування використання ресурсів: $x_{t+1}^k = a_1 x_t^k + a_2 x_{t-1}^k + ... + a_n x_{t-n+1}^k$, де t – інтервал часу, n – кількість незалежних змінних у моделі, x^k – використання ресурсу k, a – коефіцієнти моделі. На рис. 7 стовпець «+0» вказує на використання ресурсу на момент часу t, стовпець «+1» вказує на прогноз використання ресурсу на момент часу t+1, а стовпець «+2» вказує на прогноз використання ресурсу на момент часу t+2.

Під час першого запуску вхідні дані з папки даних Bitbrains завантажуються в базу даних, як показано на рис. 9. Дані бази даних будуть використовуватися в усіх подальших симуляціях.

```
Mapping Traces Data to DataBase Model
        TimeEvents created
        Reading traces
                VM 1 - done
                VM 10 - done
                VM 100 - done
                VM 1000 - done
                VM 1001 - done
                VM 1002 - done
                VM 1003 - done
                VM 1004 - done
                    • • •
                VM 997 - done
                VM 998 - done
                VM 999 - done
        Reading treaces - done |692ms|
        Reading PM capacities...6ms
Mapping Traces Data to DataBase Model - done |410ms|
Simulation runnig
        Step 1
                Updating resources requirments
                        0 VMs is finished
                        New VMs: 1114
```

Рис. 9. Завантаження вхідних даних у базу даних для подальшого моделювання

Під час симуляції застосунок відображає події, що відбувалися на кожному кроці. ПЗ також зберігає ці події у файлі журналу. На першому кроці ФС запускаються, як показано на рис. 10.

```
Simulation runnig
Step 1
        Updating resources requirments
                0 VMs is finished
                New VMs: 1083
        Updating resources requirments - done |982ms|
        Prognosing resources usage
        Prognosing resources usage - done |0ms|
        Diagnosting for overloaded servers
        Diagnostic - done |0ms|
        Assigning VMs
                Server 16 is turning on
                Server 17 is turning on
                Server 18 is turning on
                Server 19 is turning on
                Server 20 is turning on
                Server 6 is turning on
                Server 7 is turning on
                Server 8 is turning on
                Server 9 is turning on
                Server 10 is turning on
                Server 11 is turning on
                Server 12 is turning on
                Server 13 is turning on
        Assigning VMs - done |11ms|
Step 1 - finished |7ms|
```

Рис. 10. Перший крок симуляції

На рис. 11 показані результати оптимізації, а саме консолідація ВМ, за якої вдалося перемкнути три ФС (9, 10, 11) у сплячий стан. Три ВМ закінчили роботу і зупинені та одна ВМ повинна бути розміщена на якомусь ФС.

```
Step 10
        Updating resources requirments
                3 VMs is finished
                New VMs: 1
        Updating resources requirments - done |174ms|
        Prognosing resources usage
        Prognosing resources usage - done |2ms|
        Diagnosting for overloaded servers
        Diagnostic - done |0ms|
        Assigning VMs
        Assigning VMs - done |1ms|
Step 10 - finished |182ms|
Server 9 is shutting down
Server 10 is shutting down
Server 11 is shutting down
Step 11
        Updating resources requirments
                7 VMs is finished
                New VMs: 3
        Updating resources requirments - done |175ms|
```

Рис. 11. Результати оптимізації на кроці 10

План міграцій ВМ також відображається у консолі. На рис. 12 показано, що у процесі діагностики виявлено перевантажені ФС, через що був згенерований і застосований план міграцій для розвантаження ФС за допомогою модуля міграції. Отже, ФС 16 буде розвантажено у спосіб міграції віртуальних машин 280, 250 і 629 до ФС 3.

```
Step 16 - finished |267ms|
Server 3 is turning on
Step 17
        Updating resources requirments
                6 VMs is finished
                New VMs: 2
        Updating resources requirments - done |150ms|
        Prognosing resources usage
        Prognosing resources usage - done |1ms|
        Diagnosting for overloaded servers
                overloaded servers detected at prognose level 0
        Diagnostic - done |1ms|
        Migration Plan contains 3 migrations
        VM      From    To
        608     8       3
        610     8       3
        711     8       3

        Assigning VMs
        Assigning VMs - done |0ms|
Step 17 - finished |171ms|
```

Рис. 12. Згенерований план міграцій на кроці 17

6. Оцінка результатів моделювання

Під час запусків застосунку моделювання встановлено, що наявна сумарна потужність двадцяти ФС, обраних для симуляції роботи ЦОД з метою розміщення 1114 ВМ, є недостатньою, про що свідчить значна кількість міграцій ВМ.

У багатьох випадках немає необхідності визначати адаптивні порогові значення для використання певних ресурсів, оскільки експерименти показують, що агресивна по-

літика перемикання ФС у режим енергозбереження не допомагає мінімізувати кількість порушень SLA. Натомість адаптивні порогові значення рекомендується визначати відповідно до наявності стрибків робочого навантаження. Чим частіше такі стрибки, тим менше порогове значення треба використовувати. Крім того, чутливість багатьох параметрів моделі до невеликого діапазону змін істотно не впливає на показники якості.

Запропоновані методи динамічної консолідації ВМ дозволяють ефективно розвантажувати ФС, щоб перевести їх у сплячий стан, зменшуючи споживання електроенергії. Результати порівняння, представлені в [11], показують, що запропоновані методи перевершують Best Fit алгоритм щодо порушення SLA (зменшення на 46,7%), кількості активних ФС (зменшення на 33,2%) та кількості міграцій ВМ (зменшення на 42,8%).

Висновки

Останні дослідження консолідації віртуальних машин виявили необхідність дослідження нових політик управління за допомогою інструментів моделювання хмарного ЦОД з використанням реалістичного робочого навантаження.

Розроблено спеціалізоване програмне забезпечення моделювання ЦОД для реалізації підходу до динамічної консолідації віртуальних машин, що надає можливість проведення симуляцій із використанням вхідних даних, отриманих з реального ЦОД. Запропоноване програмне забезпечення дозволяє досліджувати різні режими управління ресурсами ЦОД на основі гібридного підходу консолідації віртуальних машин, який об'єднує метод управління розміщенням нових віртуальних машин та метод управління міграцією активних віртуальних машин, мінімізуючи кількість активних фізичних серверів та зменшуючи кількість порушень SLA.

У статті представлена архітектура програмного забезпечення, діаграми класів, діаграма послідовності, пакети програмного забезпечення та додаткові інструменти реалізації функцій. Експерименти показують, що запропонований програмний застосунок забезпечує широкий спектр реєстраційної та налагоджувальної інформації за допомогою текстових файлів та файлів MS Excel із діаграмами, забезпечує різноманітні види моделювання та дозволяє визначити оптимальні параметри моделі для різних режимів роботи з точки зору порушення SLA, кількості активних ФС та кількості міграцій ВМ.

References

1. IEEE Std 610.3-1989. *IEEE Standard Glossary of Modeling and Simulation Terminology*. Institute of Electrical and Electronic Engineers (IEEE), New York, NY, 1989.
2. A. Ismail, «Energy-driven cloud simulation: existing surveys, simulation supports, impacts and challenges,» *Cluster Computing*, vol. 23, pp. 3039–3055, 2020.
3. N. Mansouri, R. Ghafari, and B. Mohammad Hasani Zade, «Cloud computing simulators: A comprehensive review,» *Simulation Modelling Practice and Theory*, vol. 104, pp. 102-144,2020.
4. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23-50, 2011.
5. H. Jeon, C. Cho, S. Shin, S. Yoon, "A CloudSim-Extension for Simulating Distributed Functions-as-a-Service," *20th International Conference on parallel and distributed computing, applications and technologies (PDCAT)*, 2019, pp. 386–391.
6. B. Elahi, A. W. Malik, A. U. Rahman, M. A. Khan, "Toward scalable cloud data center simulation using high-level architecture," *Software: Practice and Experience*, vol. 50, no. 6, pp. 827–843, 2020.
7. A. Siavashi, M. Momtazpour, "GPUCloudSim: an extension of CloudSim for modeling and simulation of GPUs in cloud data centers," *The Journal of Supercomputing*, vol. 75, no. 5, pp. 2535-2561, 2019.
8. D. Oliveira, A. Brinkmann, N. Rosa, "Performability Evaluation and Optimization of Workflow Applications in Cloud Environments," *Journal of Grid Computing*, vol. 17, no. 4, pp. 749–770, 2019.
9. G. Kecskemeti, «DISSECT-CF: a simulator to foster energy-aware scheduling in infrastructure clouds,» *Simulation Modelling Practice and Theory*, vol. 58, pp. 188-218, 2015.

10. M. C. Silva Filho, R. L. Oliveira, C. C. Monteiro, P. R. Inácio, and M. M. Freire, "CloudSim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness," *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, IEEE, 2017, pp. 400-406.
11. E. Zharikov, S. Telenyk, O. Rolik, and Y. Serdiuk, "Cloud resource management with a hybrid virtual machine consolidation approach," *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*, 2019, pp. 289-294.
12. Design Patterns - Facade Pattern, [online] Available: https://www.tutorialspoint.com/design_pattern/facade_pattern.htm
13. The Repository Pattern https://msdn.microsoft.com/en-us/library/ff649690.aspx
14. .NET Framework documentation, [online] Available: https://docs.microsoft.com/en-us/dotnet/framework/
15. Interoperability Overview, [online] Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/interop/interoperability-overview
16. GWA-T-12 Bitbrains, [online] Available: http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains
17. S. Shen, V. V. Beek and A. Iosup, "Statistical Characterization of Business-Critical Workloads Hosted in Cloud Datacenters," *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Shenzhen, 2015, pp. 465-474.
18. PowerEdge Rack Servers, [online] Available: https://www.dell.com/en-us/work/shop/dell-poweredge-servers/sc/servers

Отримано 07.03.2022

***Про авторів*:**

Жаріков Едуард В'ячеславович,
доктор технічних наук,
професор кафедри інформатики
та програмної інженерії Національного
технічного університету України
«Київський політехнічний інститут
імені Ігоря Сікорського».
Кількість наукових публікацій
в українських виданнях - 125.
Кількість наукових публікацій
в зарубіжних виданнях - 37.
Індекс Гірша - 4.
http://orcid.org/0000-0003-1811-9336,

Теленик Сергій Федорович,
доктор технічних наук, професор,
декан факультету інформатики
та обчислювальної техніки Національного
технічного університету України
«Київський політехнічний інститут
імені Ігоря Сікорського».
Кількість наукових публікацій
в українських виданнях - 325.
Кількість наукових публікацій
в зарубіжних виданнях - 67.
Індекс Гірша - 4.
https://orcid.org/0000-0001-9202-9406,

Місце роботи авторів:

Національний технічний університет
України «Київський політехнічний
інститут імені Ігоря Сікорського»
Тел.: 38044 204 86 10,
e-mail: zharikov.eduard@acts.kpi.ua

УДК 004.4’24 https://doi.org/10.15407/pp2022.01.013

Я.О. Гайдукевич, А.Ю. Дорошенко

АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ ЗАПАСАМИ НА ОС ANDROID ТА БАЗИ ДАНИХ FIREBASE З ВИКОРИСТАННЯМ ШТРИХ КОДІВ ТА QR-КОДІВ

Розроблено новий програмний засіб для автоматизації запасами на основі ОС Android із використанням бази даних Firebase та мови програмування Java, що послуговується системою зчитування штрих-кодів та QR-кодів. Засіб забезпечує додавання товару на склад із подробицями про обраний товар та перегляд усіх запасів з їх ціною, категорією, назвою та кодом товару. Реалізовано авторизацію користувачів системи за допомогою відкритого стандарту FirebaseAuth та FirebaseDatabase. Створення єдиного сервісу для авторизації та реєстрації дозволило зробити систему масштабованою. У системі реалізовано пошук товару за кодом, а також перегляд запасів на складі з автоматичним підрахунком ціни. Основний сервіс бази даних – хмарна СУБД класу NoSQL, що дозволяє зберігати та синхронізувати дані між кількома клієнтами. Передбачено API для шифрування даних.

Ключові слова: ОС Android, Java, Firebase, авторизація, архітектура API, масштабовані системи автоматизації, NoSQL , СУБД , API для шифрування даних.

Вступ

Ми живемо в епоху завершення третьої цифрової революції, що почалася в другій половині минулого століття. Її характерні риси – розвиток інформаційно-комунікаційних технологій, автоматизація та роботизація виробничих процесів.

Розвиток інформаційних технологій і засобів комунікації, насамперед електронних мереж, створює потужний імпульс для формування нової тенденції функціонування сучасного бізнесу – діджиталізації економічних відносин. Більшість носіїв інформації стають цифровими, що визначає основний тренд розвитку як сучасної техніки, так і бізнес-процесів із переважною часткою електронної складової. Електронна форма комунікацій підвищує рівень і ефективність спілкування між покупцями та продавцями і створює нові ринки й можливості для реорганізації економічних процесів.

Автоматизована система управління запасами – найважливіший елемент діджиталізації підприємства. І з певного часу це вже не додаткова функція, а постійна потреба компаній, які хочуть конкурувати на ринку. Успішна діджиталізація передбачає комбінацію двох найважливіших аспектів: переведення всіх істотних активів у цифровий формат (наприклад, послуговування лише відсканованими документами, що зберігаються у хмарах) та використання спеціального програмного забезпечення для діджиталізації певних процесів компанії (наприклад, використання SAP для виконання різних транзакцій з постачальниками без урахування паперового документообігу). Цифрова трансформація - це серія загальних змін компанії, що приводить до підвищення ефективності та прибутковості компанії.

Якщо ви не знаєте, з чого почати автоматизацію вашого бізнесу, почніть із розробки індивідуальної системи управління запасами. Такий інструмент, створений безпосередньо під вашу компанію, завдяки автоматизації, допоможе м’яко перейти на електронний документообіг, водночас органічно вписуючись в існуючі процеси вашого підприємства. І пам’ятайте - завжди ухвалюйте рішення на основі даних. А ми допоможемо вам ці дані зібрати та обробити. Надалі такі системи інтегруватимуться з системами управління транспортними перевезеннями, системами автоматизації логістики запасів та закупівель. Метою даної статті є створення автоматизованої системи управління запасами на базі ОС Android для діджиталізації товарообігу підприємств.

© Я.О. Гайдукевич, А.Ю. Дорошенко, 2022

Саме через товарообіг відбувається зміна форм вартості споживчих товарів, створеної в процесі виробництва. Традиційно товарообіг досліджується переважно на мікрорівні як основний показник діяльності торговельних підприємств. Однак на сьогодні дослідження товарообігу та факторів, які на нього впливають, має відбуватися і на макрорівні, оскільки оптимальна структура товарообігу держави є одним із головних чинників конкурентоспроможності її економіки.

Запаси – це активи підприємства (ресурси, майно), які:

• утримуються для подальшого продажу (розподілу, передачі) за умов звичайної господарської діяльності;

• перебувають у процесі виробництва з метою подальшого продажу продукту виробництва;

• утримуються для споживання під час виробництва продукції, виконання робіт та надання послуг, а також управління підприємством.

До запасів належать, зокрема:

• виготовлена на підприємстві готова продукція;

• сільськогосподарська продукція і продукція лісового господарства;

• придбані (отримані) товари, що утримуються підприємством з метою подальшого продажу;

• сировина, основні й допоміжні матеріали, комплектуючі вироби та інші матеріальні цінності, призначені для виробництва продукції, виконання робіт, надання послуг, розподілу, передачі, обслуговування виробництва й адміністративних потреб.

Збільшення запасів, з одного боку, приводить до підвищення ефективності, з іншого – зростає сума коштів, зокрема, на зберігання таких запасів. Водночас, скорочення запасів може призвести до збоїв процесів виробництва, поставок чи торгівлі.

Тому важливо мати оперативну та актуальну інформацію щодо кількості запасів підприємства на складі за категоріями, цінами тощо. Зі свого боку використання програмного забезпечення для управління запасами дає можливість усьому бізнесу працювати краще. Так, зокрема, вашим співробітникам більше не буде потрібно реєструвати кожну одиницю вручну, візуально перевіряти якість і кількість отриманого товару, адже такий підхід загрожує ризиком помилок людського фактора, а також вимагає великих тимчасових витрат.

Станом на даний момент існує декілька систем, які пропонують готові рішення автоматизованих систем управління виробництвом, зокрема, Replenishment+ від AbmCloud. Основне завдання системи – забезпечити постійну наявність сировини, матеріалів, комплектуючих у потрібному місці виробничого ланцюжка у потрібній кількості та у потрібний час. А також скоротити рівень надлишкових запасів, забезпечити високу надійність поставок та скоротити вплив зміни попиту на рівень запасів. Це дуже функціонально потужний конкурент, але для управління такою системою потрібні робочі місця та комп'ютери. Тоді як система, яку розробляє автор, націлена більше на діджиталізацію. Такою системою можна керувати з мобільного девайсу в будь-якому куточку світу, сканувати товари лише з телефону та додавати до інвентарю складу.

1. Формування вимог до системи

Система має давати можливість користувачу сканувати запаси за кодом товару. Можливість додавати обрані товари до інвентарю складу за ціною, назвою, категорією товару та штрих кодом. Для кожної позиції товару користувач може скористатися пошуком за штрих кодом, щоб вивести інформацію про виріб. Для кожного нового зареєстрованого акаунту в системі передбачена база даних, яка зберігає інформацію про користувача, а також про інвентар у його системі.

Система зі свого боку також повинна автоматично підраховувати кількість одиниць товару на складі та їхню цінову складову.

Як розшифровувати інформацію, закладену в QR-кодах? Це завдання успішно вирішує переважна частина сучасних смартфонів та планшетів. Для зчитування закодованих даних досить перейти у вікно додатку товару та активувати панель сканера, QR-код буде відсканований вбудованою камерою мобільного пристрою.

Замовники з постачальниками зможуть спілкуватися та обговорювати умови закупівлі нових запасів не боячись надлишків товару на складі, адже вся інформація буде вказана в додатку.

Діаграми використання системи для таких ролей користувача, як замовник, постачальник, та спостерігач (не авторизований користувач) представлено на Рисунку 1:

Розроблена діаграма використання відображає базовий набір функцій, які доступні авторизованому користувачу автоматизованої системи управління запасами на ОС Android, а саме мерченайдзеру (товарознавець чи помічник товарознавця, людина, яка представляє виробничу чи торговельну компанію у відповідних мережах. Відповідає за викладення товару, встановлення супутнього необхідного обладнання, розміщує POS-матеріали).

Оскільки система вбудована в мобільний девайс, то функціонал дає можливість реєструвати нові товари, використовуючи лише телефон або планшет, шукати вироби за номером штрих коду. Отже, у системі укомплектовано все необхідне в єдине ціле, в одну велику мережу, де можна зберігати та вести підрахунок товарів, не використовуючи стаціонарних комп'ютерів.

Система, орієнтована на потреби торговельної компанії в процесі автоматизації управління, надає такі можливості, як реклама нового товару, перегляд власних доданих товарів до інвентарю, пошук товару за штрих кодом, видалення товару, перегляд загальної кількості товарів та цін.

Після формування варіантів використання системи, було виділено мінімальні системні вимоги до виконання додатку. Перелік атрибутів наведено у Таблиці 1:

Таблиця 1. Мінімальні системні вимоги

Операційна система	Android 4.4 (KitKat) та вище
Оперативна пам'ять	1Гб та вище
Вбудована пам'ять	8Б та вище
GPS-модуль (A-GPS)	Так
Наявність сім-картки (GPRS)	рекомендуємо
Підтримка 3G (4G, LTE)	рекомендуємо
Фото-камера	5МП та вище
Процесор	Intel Atom® Processor Z2520 1.2 ГГц або швидший процесор
Пристрої зберігання даних	Між 850 Мбайтами та 1.2 ГБ
Жорсткий диск	2 ГБ доступного місця на жорсткому диску для встановлення; додатковий вільний простір потрібен під час встановлення.

Рис. 1. Діаграма використання мерченайдзером

Під час установки APK в системі Android встановлюється файл DEX, який містить код, ресурси тощо, скомпільовані як двійкові файли. Файл dex, зазвичай, має той самий розмір, що й файл apk, якщо у вас немає ресурсів, не скомпільованих в активах. Ще однією поширеною особливістю встановлення додатків є те, що android SAVES оригінальний apk при перевстановленні у разі помилок або з якихось інших причин також встановлюється. Ось чому обсяг пам'яті наших додатків у системі вдвічі більший. Це сума розміру вихідного apk та встановленого dex.

Проєктування бази даних починається в коді AndroidStudio, де визначаються типи даних для конкретного loyaute.xml файлу в Common Attributes. Структура проєктується в класах, конкретний клас програми може відповідати за окреме вікно, таке як, наприклад, реєстрація користувача, додавання продуктів, збереження їх у базі даних тощо. Підключення бази даних до проєкту також є невід'ємною частиною. До того ж для користування бази даних Firebase треба імпортувати відповідні бібліотеки:

```
import com.google.android.gms.tasks.
OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.
FirebaseAuth;
import com.google.firebase.auth.
FirebaseUser;
import com.google.firebase.database.
FirebaseDatabase;
```

Розроблена структура бази даних включає створення змінних із відповідними типами даних, а також розгалуження на дочірні елементи. Наступний фрагмент коду демонструє додавання елемента до бази даних:

```
public void additem(){
  String itemnameValue = itemname.
getText().toString();
  String itemcategoryValue = itemcategory.
getText().toString();
  String itempriceValue = itemprice.getText().
toString();
  String itembarcodeValue = itembarcode.
getText().toString();
   final FirebaseUser users = firebaseAuth.
getCurrentUser();
  String finaluser=users.getEmail();
   String resultemail = finaluser.
replace(".","");
  if (itembarcodeValue.isEmpty()) {
  itembarcode.setError("Порожньо ");
  itembarcode.requestFocus();
  return;
  }

  if(!TextUtils.isEmpty(itemnameValue)&&!
TextUtils.isEmpty(itemcategoryValue)&&!Te
xtUtils.isEmpty(itempriceValue)){

  Items items = new Items(itemnameValue,it
emcategoryValue,itempriceValue,itembarcod
eValue);
  databaseReference.child(resultemail).
child("Items").child(itembarcodeValue).
setValue(items);
  databaseReferencecat.
child(resultemail).child("ItemByCatego
ry").child(itemcategoryValue).
child(itembarcodeValue).setValue(items);
  itemname.setText("");
  itembarcode.setText("");
  itemprice.setText("");
  itembarcode.setText("");
  Toast.makeText(additemActivity.
this,itemnameValue+" Додано",Toast.
LENGTH_SHORT).show();
  }
  else {
  Toast.makeText(additemActivity.this,"Будь
ласка, заповніть усі поля",Toast.LENGTH_
SHORT).show();
  }
}
```

Таким чином розроблений фрагмент коду додавання товарів до бази даних.

2. Реалізація бізнес-логіки

Бізнес-логіка – це частина коду, яка виконує логіку додатку. Крім бізнес-логіки, у додатках може бути код, який відповідає за відображення інформації, управління інформацією, роботу із зовнішніми ресурсами/сервісами.

У системі бізнес-логіка зосереджена в класах, які синхронізовані з базою даних, що зберігає інформацію у форматі обміну

даних JSON. Крім цього при запитах на модифікацію даних передаються відповідні заголовки авторизації для уникнення несанкціонованого доступу.

Нижче наведено декілька реалізацій класів.

Клас scanItemsActivity призначений для роботи зі штрих кодами та QR кодами, наприклад, пошук товару за номером коду. Клас утворює два вікна: одне – пошук товару за кодом, а інше – вивід товару з параметрами, такими як: назва виробу, його код, ціна і категорія.

Також викликається збережена процедура, яка збирає із бази даних інформацію та повертає результат у форматі JSON. Клас сервісу складається з наступних методів:

1. метод firebasesearch. Метод створює запит до бази даних, щодо номера коду товару, який має повернути або вивести на екран, а також створює об'єкт FirebaseRecyclerAdapter, який зв'язується з Query до RecyclerView. Коли дані додаються, видаляються або змінюються, ці оновлення автоматично застосовуються до інтерфейсу користувача в реальному часі;

```
        public void
firebasesearch(String searchtext){
        Query firebaseSearchQuery
= mdatabaseReference.
orderByChild("itembarcode").
startAt(searchtext).
endAt(searchtext+"\uf8ff");

FirebaseRecyclerAdapter<Items,
        UsersViewHolder>
firebaseRecyclerAdapter = new
FirebaseRecyclerAdapter<Items,
UsersViewHolder>
         ( Items.class,
         R.layout.list_layout,
         UsersViewHolder.class,
         firebaseSearchQuery )
         {
```

2. метод setDetails. Метод створює TextView для виводу їх на екран додатку після пошуку конкретного товару, а саме код товару, назву, категорію та ціну.

За аналогічною схемою реалізовано клас (viewInventoryActivity) для роботи з інвентарем складу, але виводяться всі товари без пошуку.

```
        public void setDetails(Context
ctx,String itembarcode, String
itemcategory, String itemname, String
itemprice){
        TextView item_barcode =
(TextView) mView.findViewById(R.
id.viewitembarcode);
        TextView item_name =
(TextView) mView.findViewById(R.
id.viewitemname);
        TextView item_category =
(TextView) mView.findViewById(R.
id.viewitemcategory);
        TextView item_price =
(TextView) mView.findViewById(R.
id.viewitemprice);

        item_barcode.
setText(itembarcode);
        item_category.
setText(itemcategory);
        item_name.setText(itemname);
        item_price.setText(itemprice);
         }
```

3. Метод onCreate. Метод працює з базою даних, створює точку входу для доступу до бази даних Firebase. Можливість отримати екземпляр, використавши getInstance(). Щоб отримати доступ до розташування в базі даних та прочитати або записати дані, створено getReference().

```
        @Override
         protected void
onCreate(Bundle savedInstanceState)
{
         super.
onCreate(savedInstanceState);
         setContentView(R.layout.
activity_scan_items);
         firebaseAuth = FirebaseAuth.
getInstance();
         final FirebaseUser users =
firebaseAuth.getCurrentUser();
```

```
        String finaluser=users.
getEmail();
        String resultemail = finaluser.
replace(“.”,””);
        mdatabaseReference =
FirebaseDatabase.getInstance().
getReference(“Users”).
child(resultemail).child(“Items”);
        resultsearcheview =
findViewById(R.id.searchfield);
        scantosearch
= findViewById(R.
id.imageButtonsearch);
        searchbtn = findViewById(R.
id.searchbtnn);

        mrecyclerview =
findViewById(R.id.recyclerViews);
        LinearLayoutManager
manager = new
LinearLayoutManager(this);
        mrecyclerview.
setLayoutManager(manager);
        mrecyclerview.
setHasFixedSize(true);

        mrecyclerview.
setLayoutManager(new
LinearLayoutManager(this));
        }
```

Клас RegisterActivity призначений для роботи із користувачами системи управління запасами, реєстрації їхніх акаунтів та збереження у базі даних, щоб зберігати тільки ту інформацію про товар, яку додавав конкретний користувач. Конфіденційність даних у базі присутня, тому паролі не виводяться задля безпеки даних. Клас складається з наступних методів:

1. метод onCreate. Містить логіку реєстрації користувача, створює текстові поля для заповнення даних, таких як емейл, пароль, повтор пароля та ім’я . Щоб отримати об’єкт FirebaseAuth, викликає статичний метод getInstance().

```
    @Override
        protected void
onCreate(Bundle savedInstanceState)
{
        super.
onCreate(savedInstanceState);
        setContentView(R.layout.
activity_register);

        editTextName =
findViewById(R.id.departmentName);
        editTextEmail =
findViewById(R.id.emailRegister);
        editTextPassword
= findViewById(R.
id.passwordRegister);
        editTextcPassword=
findViewById(R.id.confirmPassword);
        UserRegisterBtn=
findViewById(R.id.button_register);
        progressBar =
findViewById(R.id.progressbar);
        progressBar.
setVisibility(View.GONE);
        mAuth = FirebaseAuth.
getInstance();

        UserRegisterBtn.
setOnClickListener(new View.
OnClickListener() {
        @Override
        public void onClick(View v) {
        registerUser();
        }
        });
        }
```

2. метод onStart. Обробляє вже зареєстрованого користувача.
3. метод registerUser. Обробляє поля, що були у методі onCreate, надає їм тип даних .toString(), Якщо поля неправильно заповнені або пусті видає відповідне повідомлення. Метод отримує дані та надсилає дані на основі електронної пошти користувача до бази даних, а також виводить повідомлення у разі успішної або неуспішної реєстрації.

Клас additemActivity потрібен для додавання товару до інвентарю складу. Також він створює поля для вводу даних про вибір, таких як: назва товару, категорія, ціна та числовий тип даних для вводу номера коду товару. Крім того, клас від-

слідковує зміну в полях під час зберігання товару та додає до бази даних. Клас складається з наступних методів:

1. метод onCreate. Містить логіку реєстрації товару, створює текстові поля для заповнення даних, таких як: назва товару, категорія, ціна та числовий тип даних для вводу номера коду товару. Щоб отримати об'єкт FirebaseAuth, викликається статичний метод getInstance(). Нижче наведено реалізацію:

```
@Override
  protected void
onCreate(Bundle savedInstanceState)
{
    super.
onCreate(savedInstanceState);
    setContentView(R.layout.
activity_additem);
    firebaseAuth = FirebaseAuth.
getInstance();

    databaseReference =
FirebaseDatabase.getInstance().
getReference("Users");
    databaseReferencecat =
FirebaseDatabase.getInstance().
getReference("Users");
    resulttextview =
findViewById(R.id.barcodeview);
    additemtodatabase
= findViewById(R.
id.additembuttontodatabase);
    scanbutton =
findViewById(R.id.buttonscan);
    itemname = findViewById(R.
id.edititemname);
    itemcategory=
findViewById(R.id.editcategory);
    itemprice = findViewById(R.
id.editprice);
    itembarcode=
findViewById(R.id.barcodeview);


    scanbutton.
setOnClickListener(new View.
OnClickListener() {
    @Override
    public void onClick(View
view) {
      startActivity(new
Intent(getApplicationContext(),
ScanCodeActivity.class));
      }
      });

      additemtodatabase.
setOnClickListener(new View.
OnClickListener() {
      @Override
      public void onClick(View v) {
      additem();
      }
      });
    }
```

2. метод additem. Виконує такі функції, як: додавання елемента до бази даних, створення полів для їх заповнення, передавання цих полів до бази даних, зберігання.
3. метод Logout. Потрібен для виклику меню у верхній частині екрану та виходу з акаунту. Цей метод фігурує також у інших класах, де є можливість виклику меню. Наведено реалізацію нижче:

```
    private void Logout()
{
 firebaseAuth.signOut();
 finish();
 startActivity(new
Intent(additemActivity.
this,LoginActivity.class));
  Toast.makeText(additemActivity.
this,"LOGOUT SUCCESSFUL",
Toast.LENGTH_SHORT).show();
}
```

4. Метод onCreateOptionsMenu. Потрібен для MenuInflater - це системний ресурс Android. Створюється під час завантаження андроїда. Це постійний об'єкт, і посилання на нього завжди доступне у пам'яті. Кожен підклас класу Context, тобто Activity може отримати посилання на нього, викликавши getMenuInflater()зсередини класу. Наведено нижче реалізацію:

```
        @Override
public boolean
onCreateOptionsMenu(Menu menu) {
  getMenuInflater().inflate(R.menu.
menu,menu);
  return true;
}
```

Клас ScanCodeActivity. Відповідає за сканування штрих кодів та QR кодів. Нижче наведено реалізацію (використано бібліотеку ZXingScannerView scannerView):

```
        import com.google.zxing.
Result;
        import me.dm7.
barcodescanner.zxing.
ZXingScannerView;

        public class ScanCodeActivity
extends AppCompatActivity
implements ZXingScannerView.
ResultHandler {
         int MY_PERMISSIONS_
REQUEST_CAMERA=0;

         ZXingScannerView
scannerView;
         @Override
         protected void
onCreate(Bundle savedInstanceState)
{
         super.
onCreate(savedInstanceState);
         scannerView = new
ZXingScannerView(this);

setContentView(scannerView);
         }

         @Override
         public void
handleResult(Result result) {

         additemActivity.
resulttextview.setText(result.
getText());

         onBackPressed();
         }

         @Override
         protected void onPause() {
         super.onPause();
         scannerView.stopCamera();
         }

         @Override
         protected void
onPostResume() {
         super.onPostResume();
         if (ContextCompat.checkSelf
Permission(getApplicationContext(),
Manifest.permission.CAMERA)
         != PackageManager.
PERMISSION_GRANTED)
         {
         ActivityCompat.
requestPermissions(this, new String[]
{Manifest.permission.CAMERA},
         MY_PERMISSIONS_
REQUEST_CAMERA);
         }
         scannerView.
setResultHandler(this);
         scannerView.startCamera();
}}
```

Таким чином було реалізовано панель сканування для штрих кодів та QR кодів.

3. Тестування

У структурі Android Studio є папки androidTest і test поруч з основною папкою з класами проєкту.

Навіщо потрібне тестування? Якщо додаток маленький, то тести не потрібні й цілком можна обходитися без тестування і далі. Чому? Йдеться про те, що в невеликих проєктах можна контролювати логіку програми. Також можна передбачити слабкі місця та виправити код.

Але все змінюється, якщо програма стала складною. Якщо з'явилося понад десятка різних екранів активностей, окремих класів тощо, код слід розбивати на модулі, аби забезпечити незалежність. Такий підхід обов'язково використовується у компаніях, де кожен відповідає за свою ділянку коду.

Тести поділяються на дві категорії – локальні (Unit Testing) та інструментальні (UI Testing).

Локальні тести перевіряють роботу методу, класу, компонента. Тест не залежить від Android. Ппрактично, перевіряється код Java, який можна контролювати на звичайному комп'ютері без участі пристрою або емулятора. Наприклад, такому варіанту відповідає додавання двох чисел типу int. Подібні тести проводять у папці Test.

Для інструментальних тестів наявність пристрою або емулятора є обов'язковою, оскільки потрібно тестувати натискання кнопки, введення тексту, прокручування, торкання та інші операції. Тести проводять у папці androidTest.

Висновки

Реалізовано програмний засіб для автоматизації запасами на основі ОС Android із використанням бази даних Firebase та мови програмування Java. Система використовує метод зчитування штрих кодів та QR-кодів. Він забезпечує додавання товару на склад з усіма подробицями про обраний товар та перегляд усіх запасів на складі з їхньою ціною, категорією, назвою та кодом товару. Реалізовано авторизацію користувачів системи за допомогою відкритого стандарту FirebaseAuth та FirebaseDatabase.

Надано можливість реєструвати користувачів системи, що дозволило зробити систему масштабованою. У системі реалізовано пошук товару за кодом, а також перегляд запасів на складі з автоматичним підрахунком ціни. Основний сервіс бази даних - хмарна СУБД класу NoSQL, що дозволяє зберігати та синхронізувати дані між кількома клієнтами. Передбачено API для шифрування даних. Система у грошовому еквіваленті автоматично розраховує результати управління запасами товарів та загальну ціну.

Бібліотека me.dm7.barcodescanner.zxing.ZXingScannerView використовувалась для реалізації сканера.

Запропонована автоматизована система управління запасами уможливлює обрання найбільш ефективних інструментів для регулювання товарообігу та управління запасами.

References

1. Stateless 3.0 - A State Machine library for .NET Core [Online] - Access mode: https://www.hanselman.com/blog/stateless- 30-a-state-machine-library-for-net-core /.
2. Problems of information technology software development for supercomputer systems [Online]. - Access mode https://zakon.rada.gov.ua/rada/show/v0347550-10#Text.
3. Architectural Styles and the Design of Network-based Software Architectures [Online] - Access mode: https://www.ics.uci.edu/~fielding/pubs/dissertation / top.htm
4. Alexander Wald, Paul Datel, Harvey Datel. Android for developers. 3rd edition, 2016
5. John Horton. Learning Java by Building Android Games - Explore Java Through Mobile Game Development, 2019.
6. Greg Nudelman. Android Design Patterns: Interaction Design Solutions for Developers, 2013.
7. Mark L. Murphy. Busy Coder's Guide to Android Development, 2020.
8. John Darwin. Android. Collection of recipes, 2018.
9. Don Griffiths, David Griffiths. Head First. Programming for Android, 2016.
10. Robert Cecil Martin. Net code - 2008.
11. Christine Marsicano, K. Stewart, Bill PhillipsAndroid. Programming for Professionals, 4th Edition, 2021.

Література

1. Stateless 3.0 – бібліотека State Machine для .NET Core [Online] – Режим доступу: https://www.hanselman.com/blog/stateless-30-a-state-machine-library-for-net-core/ .
2. Проблеми розробки програмного забезпечення інформаційних технологій для суперкомп'ютерних систем [Online] – Режим доступу:https://zakon.rada.gov.ua/rada/show/v0347550-10#Text.
3. Архітектурні стилі та дизайн архітектур програмного забезпечення на основі мережі [Online] – Режим доступу: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.
4. Олександр Уолд, Пол Датель, Харві Датель. Android для розробників. 3-е видання, 2016.
5. Джон Хортон. Вивчення Java шляхом створення ігор для Android – Досліджуйте Java за допомогою розробки мобільних ігор, 2019.

6. Грег Нудельман. Шаблони дизайну Android: рішення для дизайну взаємодії для розробників, 2013.
7. Марк Л. Мерфі. Посібник Busy Coder з розробки Android, 2020.
8. Джон Дарвін. Android. Збірник рецептів, 2018.
9. Дон Гріффітс, Девід Гріффітс. Голова спочатку. Програмування для андроїд, 2016.
10. Роберт Сесіл Мартін. Чистий код – 2008 р.
11. Крістін Марсікано, К. Стюарт, Білл ФілліпсAndroid. Програмування для професіоналів, 4-е видання, 2021.

Отримано 18.12.2021

***Про авторів*:**

Гайдукевич Ярослав Олегович,
магістрант кафедри інформаційних систем та технологій КПІ імені Ігоря Сікорського. Кількість наукових публікацій в українських виданнях – 1. http://orcid.org/0000-0002-6300-1778,

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук, професор, завідувач відділу теорії комп'ютерних обчислень Інституту програмних систем НАН України, професор кафедри автоматики і управління в технічних системах НТУУ «КПІ імені Ігоря Сікорського». Кількість наукових публікацій в українських виданнях – понад 180. Кількість наукових публікацій в іноземних виданнях – понад 70.
Індекс Гірша – 6.
http://orcid.org/0000-0002-8435-1451

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559.
e-mail: doroshenkoanatoliy2@gmail.com, yarmcfly@gmail.com

УДК 681.3 https://doi.org/10.15407/pp2022.01.23

Ю.В. Рогушина

ВИКОРИСТАННЯ СИСТЕМ ОРГАНІЗАЦІЇ ЗНАНЬ НА ОСНОВІ ОНТОЛОГІЙ У WIKI-РЕСУРСАХ

У статті розглядаються теоретичні основи систем організації знань (СОЗ) в інтелектуальних застосуваннях на основі онтологій. Ціллю даного дослідження є аналіз застосування різних типів СОЗ для організації та вдосконалення бази знань семантизованих Wiki-ресурсів, які містять гетерогенний мультимедійний контент великого обсягу та мають складну структуру, що інтегрує знання із різних ПрО. Розглянуто діалекти мови подання онтологій OWL та їх виразність для подання окремих випадків онтологій, що використовуються у СОЗ. Проаналізовано критерії класифікації СОЗ та сфери їх застосування. Запропоновано формальну модель онтології семантизованого Wiki-ресурсу та засоби реалізації різних видів відношень між об'єктами у середовищі Semantic MediaWiki з використанням шаблонів, розглядаються проблеми доступу до інформації у цих ресурсах з точки зору СОЗ та наводяться методи й засоби вирішення цих проблем. Представлено реалізацію запропонованого підходу на прикладі портальної версії Великої Української Енциклопедії (е-ВУЕ).

Ключові слова: система організації знань, онтологія, тезаурус, Wiki-ресурс.

Вступ

Характерною рисою сучасних інтелектуальних інформаційних систем (ІІС) є використання знань – як внутрішніх, або зі заздалегідь визначених джерел, так і зовнішніх, які генеруються іншими ІІС, або створюються на основі аналізу зовнішніх інформаційних ресурсів (ІР) різного рівня структурування. На ефективність роботи ІІС впливають як вибір форми та організації подання знань, що обробляються, так і джерела знань і методи їх здобуття. Сьогодні більшість Web-орієнтованих ІІС використовують онтології різної складності та розміру. Крім того, все частіше ІІС застосовують засоби інтеграції з великими даними (Big Data) для побудови на їх основі необхідних закономірностей та правил. Тому велике значення мають ті системи організації знань (СОЗ), що використовуються як концептуальна інфраструктура для підтримки цього процесу.

СОЗ забезпечують розуміння, інтеграцію та пошук знань, підготовку знань до застосування, надають можливості для виявлення нових зв'язків і узагальнень, для прогнозування, формулювання нових гіпотез та прийняття рішень на їх основі. Важливо розуміти характеристики, які можна використовувати для опису та аналізу СОЗ. Ці характеристики поділяються на внутрішні, що характеризують типи та властивості знань у системі (наприклад, які підкласи онтологій застосовуються, які відношення між поняттями підтримуються), і зовнішні, що не стосуються внутрішньої природи СОЗ, але описують джерела та засоби її поповнення (наприклад, автоматизовані або ручні).

Онтологічне подання знань

Грубер визначає онтології як явні специфікації концептуалізації [2]. Онтології базуються на поданні знань як скінчнної множини об'єктів (класів та індивідів) [3], що називається інтерпретацією понять. Як правило, онтології можна представити у вигляді орієнтованих графів, вузли яких представляють поняття предметної області (ПрО), а ребра – відношення між цими поняттями. Різні структури знань, що базуються на онтологіях, відрізняються за типами відношень між поняттями, за своїми властивостями і різними логічними характеристиками цих властивостей.

Формальну модель онтології O у найбільш узагальненому вигляді часто

© Ю.В. Рогушина, 2022

представляють у вигляді впорядкованої трійки: O = < X, R, F > (1), де T – скінченна множина понять ПрО, яку представляє онтологія O; R – скінченна множина відношень між поняттями цієї ПрО; F – скінченна множина аксіом і функцій інтерпретації понять і відношень онтології O. Відношення визначають собою тип взаємодії між поняттями. Аксіоми використовуються для моделювання тверджень, які завжди є істинними для ПрО. Ця формальна модель може бути уточнена відповідно до цілей розвитку онтології. Наприклад, деякі дослідження уточнюють X як X = T ∪ P, де T — скінченний набір понять предметної області, а P — скінченний набір властивостей понять. Інші поділяють X підмножини класів і екземплярів класів. Особливі випадки онтології можуть бути визначені специфікаціями та деякими обмеженнями на X, R і F та їх підкласи. Окремі випадки онтологій, таких як глосарій, таксономія, каталог тощо, можуть бути визначені обмеженнями та специфікаціями моделі (1), які визначають можливі елементи X, R і F. Детальніше формальні моделі онтологій досліджено в [4].

Для практичного використання онтологій як джерела знань щодо ПрО потрібно забезпечити їх широкомасштабну інтероперабельність та формалізоване спільне розуміння. Тому для представлення онтологій доцільно використовувати мови OWL (Web Ontology Language) та RDF, які розроблені Консорціумом World Wide Web (W3C) в рамках проєкту Semantic Web. Semantic Web надає засоби для перетворення Web на глобальну базу знань, що забезпечує взаємодію між системами через обмін даними та пошук інформації на рівні знань. OWL та RDF можна обробляти за допомогою мови запитів RDF SPARQL, яка забезпечує доступ до онтологічних знань, що містяться в них.

Мова подання онтологій OWL розширює можливості XML, RDF, RDF Schema та DAML+OIL. Онтологія OWL є послідовністю аксіом, фактів і посилань на інші онтології, а також компоненти для запису авторства та іншої подібної інформації. Онтології OWL є документами Web, на які можна посилатися через URI. Онтології OWL зазвичай містять: 1. Класи, що визначаються у owl:Class; 2. Екземпляри, що визначаються у owl:Thing; 3. Властивості, що визначаються у owl:ObjectProperty та owl:DatatypeProperty; 4. Правила – твердження, які застосовуються для логічного виведення.

OWL має три діалекти, що різняться за виразністю та складністю обробки: OWL Lite; OWL DL; OWL Full. OWL Lite – найпростіший варіант, призначений для тих користувачів, які мають потребу класифікувати ієрархію й використовують прості обмеження. OWL Lite забезпечує швидку міграцію тезаурусів та інших таксономій. OWL DL є розширенням OWL Lite, а OWL Full – розширенням OWL DL. Як наслідок, будь-яка онтологія OWL Lite є онтологією OWL DL, а будь-яка онтологія OWL DL є онтологією OWL Full.

OWL DL орієнтований на тих користувачів, які потребують максимальної виразності без втрати повноти обчислень і гарантованого завершення всіх обчислень у визначений час. OWL DL містить усі мовні конструкції OWL з обмеженнями поділу типу (клас не може бути окремою властивістю, а властивість – індивідом або класом). Назва OWL DL пов'язана з його відповідністю дескриптивній логіці.

OWL Full призначається для користувачів, яким потрібна максимальна виразність і синтаксична потужність RDF без обчислювальних гарантій. Наприклад, у OWL Full клас може одночасно розглядатися і як сукупність екземплярів, і як екземпляр. Інша суттєва відмінність від OWL DL у тому, що owl:DatatypeProperty може бути позначена як owl:InverseFunctionalProperty. OWL Full припускає такі онтології, що розширюють склад визначеного словника RDF або OWL.

У виборі діалекту OWL для створення онтології потрібно проаналізувати, які саме виразні можливості необхідні для коректного відображення знань щодо ПрО, та обирати менш складний з при-

датних. Наприклад, для подання тезаурусів доцільно застосовувати OWL Lite.

Мова OWL широко використовується у Web-орієнтованих ІІС, і в процесі її практичного застосування виявилися певні обмеженість її виразних можливостей і недоліки технічного характеру (складність синтаксичного розбору, неможливість знайти помилки в іменах), які спричиняють багато проблем у створенні прикладних систем організацій знань. Це викликало потребу створення нової версії мови – OWL 2.0.

Якщо в мові OWL можна визначати лише симетричні й транзитивні властивості, то OWL 2. 0 дає змогу розширити спектр логічних характеристик властивостей рефлексивністю, антирефлексивністю й антисиметричністю. Також додається можливість декларування локальної рефлексивності, яка використовується тоді, коли для властивості рефлексивність не характерна, а для деяких класів об'єктів рефлексивність наявна. Якщо OWL давав змогу лише позначити класи як несумісні, то в OWL 2. 0 з'явилася можливість робити те саме й для властивостей. Оголошення деякої множини властивостей несумісними означає, що два екземпляри не можуть бути з'єднані більш ніж однією властивістю з цієї множини (приміром, два об'єкти можуть бути пов'язані або властивістю «знаходитися над», або властивістю «знаходитися всередині», але не обома одночасно).

OWL 2. 0 містить спеціальну конструкцію для визначення ключа – множини властивостей, яка дає змогу унікально ідентифікувати екземпляри заданого класу. Також в OWL 2. 0 розширено набір типів даних для властивостей атрибутів і введено можливість задавати деякі власні обмеження на діапазони значень. Ще одна важлива риса OWL 2. 0 – визначення нових властивостей через композицію інших властивостей (property chain inclusion).

Окремі випадки онтологій та їхні формальні моделі

Онтології є досить складним для обробки засобом подання знань, і тому для розв'язку багатьох практичних завдань використовують їх окремі випадки, що містять різноманітні обмеження та припускають простіші методи аналізу.

Каталог – це сукупність понять X без формальних зв'язків між ними: $R = \varnothing$.

Семантика понять може бути описати анотаціями природної мови (ПМ). На відміну від відношень і властивостей, такі анотації припускають неоднозначні інтерпретації. Каталоги можна розглядати як найпростіший окремий випадок онтології, що представлений набором понять, котрі є екземплярами одного класу з єдиною властивістю "Анотація", і порожнім набором відношень.

Глосарій – це впорядкований (наприклад, за абеткою) набір понять ПрО з визначеннями (анотаціями) цих термінів. Це окремий випадок онтології з набором відношень, що містить єдине значення R = {"next"}. Анотація поняття є єдиною властивістю екземплярів єдиного класу [5]

Таксономія – це схема ієрархічної класифікації, де поняття організовані в групи або типи: R = {"subclass"}. Таксономії можна використовувати для організації та індексації знань щодо документів, статей, відео тощо. Таксономія – це окремий випадок онтології з єдиним ієрархічним відношенням, яке має такі характеристики, як транзитивність та антирефлексивність.

*Легка (*lightweight) *онтологія* — це онтологія, де поняття пов'язані загальними асоціаціями, а не строго визначеними формальними зв'язками. Досить часто легкі онтології розглядають як онтології, що складаються лише з набору базових таксономій. Тобто R містить кілька різних відношень типу "клас-підклас". Термін «легка онтологія» широко використовується для позначення простих таксономій понять, організованих ієрархічно для семантичної взаємодії щодо термінології з групами користувачів. Проте деякі дослідники [6] розширюють це поняття, узагальнюють відношення «є частиною» до понять, що відповідають основним властивостям базових таксономій. Тобто у легкій онтології розширення поняття дочірнього вузла є підмножиною розширення концепції батьківського вузла.

Ширше визначення розглядає легкі онтології як онтології із обмеженими наборами відношень між концептами. Саме цим вони відрізняються від виразніших важких онтологій [7], але явно не визначається, які саме обмеження можуть за таких умов застосовуватися. В цьому випадку формальна модель легкої онтології — це підтип трійки (1), де <X, R> — кореневе дерево, а F — скінченний набір понять, виражених формальною мовою F, що належить до сімейства пропозиційної логіки опису (DL) мови без ролей. Кожну концепцію легких онтологій можна перевести у вираз DL. Приклади спрощених онтологій включають асоціативну мережу та багатомовні класифікації, але цей термін не використовується послідовно. В своїх попередніх дослідженнях [8] ми вважали, що онтологія є легкою, якщо для неї характерна наявність тільки наступних відношень:

- відношення "клас-підклас", значення яких пов'язані транзитивно;
- об'єктні відношення синонімії, значення яких пов'язані симетрично;
- об'єктні відношення, що специфічні для ПрО, значення яких не мають бінарних властивостей.

Водночас аксіоми та правила в легкій онтології не застосовуються: $F = \varnothing$.

Таксономії, тезауруси, бізнес-каталоги, фасетні класифікації, Web-каталоги та класифікації користувачів можна розглядати як неформальні прототипи формальних легких онтологій. Як показано в [9], формальні легкі онтології можуть бути автоматично створені з неформальних класифікацій користувачів.

Легкі онтології поділяють за їх використанням на два основні типи: 1. *описові* легкі онтології: для визначення значення термінів, а також природи та структури домену; 2. легкі онтології *класифікації*: для опису, класифікації та доступу до великих колекцій документів або даних.

Легкі онтології можуть використовуватися для інтеграції даних. Цей процес об'єднує дані з різних джерел і надає користувачеві можливість уніфікованого погляду на ці дані. Часто таке джерело даних може бути представлене у вигляді кореневого дерева ("rooted tree"), де вузли пов'язані з поняттями та їхніми природномовними мітками, а інтеграція даних може бути полегшена виявленням семантичних відношень між цими поняттями на основі даних. Знайдене семантичне відношення між двома вузлами може бути класифіковано як ієрархічне, еквівалентне або відношення неперетину. Такі відношення потім можуть бути використані для інтеграції різних СОЗ. У легких онтологіях семантичні відношення можуть бути визначені між елементами контрольованих словників, таксономій, тезаурусів, бізнес-каталогів, фасетних класифікацій тощо.

Системи класифікації та таксономії можуть бути перетворені на формальні системи, що описуються конструкціями DL замість природномовних текстів, які допускають неоднозначну та суб'єктивну інтерпретацію. Таксономії, тезауруси, Web-каталоги тощо можна використовувати як прототипи формальних легких онтологій, створення яких потребує участі експертів ПрО [10]. Будь-яка описова спрощена онтологія може бути використана як класифікаційна легка онтологія, але не навпаки.

Більш складні онтології надають ширші виразні засоби для подання знань і можуть використовуватися для ПрО з різноманітними специфічними властивостями, які не можуть бути представлені ієрархічними («клас-підклас») і мереологічними («є частиною») відношеннями [11]. Але обробка складних онтологічних структур потребує значно більше обчислювальних ресурсів і часу. Тому в практичних застосуваннях виникає потреба в їх редукції відповідно до вимог задач, але без втрати необхідних знань.

Онтології в СОЗ

Інтелектуальні програми використовують СОЗ різних типів (такі як схеми класифікації, тезауруси, тематичні карти, онтології) та різного обсягу. Термін «системи організації знань» позначає групу засобів, спрямованих на упорядкування інформації та підтримку управління зна-

ннями. Зазвичай, такі системи забезпечують ефективніший пошук та збереження знань у певному ІР.

СОЗ є інструментами для опису контенту ІР і допомоги в доступі й пошуку документів та інформації [12]. У вузькому значенні КОС підтримують такі різноманітні види діяльності, як опис документів, індексація та класифікація в бібліотеках, архівах, бібліографічних базах даних тощо. У ширшому значенні такі системи використовуються для організації науково-освітніх установ, структури дисциплін та професій, поширення знань тощо. СОЗ може використовуватися як міст між інформаційними потребами користувачів та контентом ІР. Основні елементи більшості СОЗ можуть бути виражені в RDF за допомогою Simple Knowledge Organization System (SKOS) [13]. СОЗ забезпечує ідентифікацію тих інформаційних об'єктів (ІО), що можуть зацікавити користувачів, із використанням деяких додаткових знань щодо користувача. Серед СОЗ виділяють чотири основні групи, які можуть перетинатися:

- списки термінів;
- моделі, подібні до метаданих;
- класифікація та категоризація;
- моделі відношень.

Списки термінів містять:

- Списки: впорядковані скінченні множини термінів;

- Словники: алфавітні списки понять з варіантами визначень (зазвичай представлені природномовним текстом);

- Глосарії: алфавітні списки понять з єдиним визначенням для кожного відповідно до ПрО;

- Кільця синонімів: набори понять, які вважаються еквівалентними для пошуку в певній ПрО.

Моделі, подібні до метаданих ("Metadata-like Models") містять:

- нормативні файли ("Authority Files"): списки понять, які використовуються для керування варіантами термінів для об'єктів в обраній ПрО;

- директорії ("Directories"): списки імен і деяка контактна інформація, пов'язана із цими іменами;

- географічні довідники ("Gazetteers"): індекси із геопросторовими словниками назв та географічних об'єктів.

Класифікація та категоризація містять:

- Предметні рубрики ("Subject Headings"): схеми зі скінченним набором контрольованих понять для представлення тем (рубрик) для елементів колекції та набори правил, які об'єднують ці поняття в структуровані заголовки;

- Схеми категоризації: неформальні схеми для групування;

- Таксономії: класифікація предметів на групи або категорії на основі деяких обраних властивостей;

- Схеми класифікації: ієрархічна та фасетна систематизація кількісних або алфавітних позначень для подання теми.

Моделі відношень включають більш складні СОЗ:

- Тезауруси: скінченні набори відношень понять між термінами, що чітко відображені та ідентифіковані через стандартизовані відношення (включаючи відношення ієрархії, еквівалентності та асоціації);

- Семантичні мережі: множини понять, представлені вузлами мережі і з'єднані дугами, які представляють відношення між поняттями;

- Онтології: концептуальні моделі, що представляють різноманітні складні відношення між об'єктами, включаючи правила та аксіоми, відсутні в семантичних мережах.

Критерії класифікації СОЗ залежать від цілі такого дослідження. За такий критерій часто використовують семантичну потужність, яка визначається як набір семантичних відношень між поняттями (класами та екземплярами), що підтримуються в СОЗ.

Типи онтологій, які використовуються в СОЗ, проаналізовано в [14]. Вони представлені широким спектром артефактів, що задовольняють визначенню онтології Грубером [2]. Онтології, що використовуються в СОЗ, також можна класифікувати за багатьма параметрами. СОЗ можна класифікувати також відповідно до їхньої структури та функцій:

структура може варіюватися від пласкої до двовимірної та багатовимірної, а їхні функції можуть містити усунення неоднозначностей, контроль синонімів, встановлення ієрархічних та асоціативних зв'язків та представлення властивостей. Прості пласкі структури СОЗ представлені списками вибору та кільцями синонімів. Приклади двовимірних структур СОЗ використовують ієрархії, а багатовимірні структури СОЗ використовуються семантичними мережами на основі різних семантичних типів та онтологій.

Таксономія СОЗ, запропонована в [15], базується на тому, які з основних типів відношень між поняттями в них підтримуються:

- контрольовані синоніми;
- ієрархічні відношення;
- асоціативні відношення;
- представлення властивостей.

У твердженнях природної мови досить часто виникає неоднозначність, якщо слово або словосполучення мають більше одного значення. СОЗ надають різні способи усунення неоднозначності. Якщо СОЗ не підтримує відношення між поняттями, то можуть використовуватися досить прості способи уточнення семантики цих понять. Один із них заснований на використанні доменного імені для уточнення поняття. Наприклад, «Меркурій (метал)», «Меркурій (планета)» замість «Меркурій». Такий підхід широко використовується в онлайн-енциклопедіях і Wiki-словниках для подання багатозначних термінів. Інший підхід уточнює значення поняття, надаючи контекст для нього, наприклад, за допомогою «списку вибору» – скінченного впорядкованого (за алфавітом, хронологічно, географічно тощо) набору понять з однієї групи. Такі списки описують об'єкти класів з обмеженою кількістю елементів. Списки можна ефективно використовувати для перегляду та пошуку. Вони часто використовуються як найпростіший спосіб структурування та створення метаданих.

Контрольовані синоніми (еквіваленти) використовуються для прийняття рішення в ситуаціях, коли поняття представлено більш ніж одним терміном (має близькі синоніми), тобто його можна описати відмінною, але змістовно еквівалентною термінологією. Набір синонімів може містити терміни різних мов, акроніми та варіанти написання. Найпоширеніші проблеми стосуються близьких синонімів, значення яких зазвичай визначаються як різні, але можуть розглядатися як еквівалентні для певної ПрО. Наприклад, «машина» є синонімом «автомобіль» у ПрО транспорту, але у сфері інформатики це поняття є синонімом для «комп'ютер» .

Ієрархічні відношення є найбільш поширеними відношеннями в СОЗ. Використання ієрархічних зв'язків розглядається як основний критерій для виокремлення таксономій і тезаурусів від простіших форм СОЗ, таких як списки та кільця синонімів. Ці відношення визначають рівні підпорядкування, які поділяють клас на підкласи, де кожен підклас є підмножиною вихідного класу. Класи одного рівня об'єднуються в класи вищого рівня.

Ієрархічні відношення охоплюють три різні групи відношень: 1. родові відношення ("клас-підклас"); 2. інстанційні відношення ("екземпляр-клас"); 3. мереологічні відношення ("ціле-частина").

Родові відношення визначають зв'язки між класом і його підкласами. Цей тип відношення може пов'язувати деякий вужчий термін з більш широким терміном.

Інстанційні відношення визначають зв'язок між загальною категорією предметів чи процесів і окремим екземпляром цієї категорії. Цю групу відношень можна узагальнити виразом природної мови «приклад».

Мереологічні відношення охоплюють ситуації, коли одні поняття за своєю природою включені в інші, так що поняття можна організувати в логічні ієрархії. Усі ці відношення можна задати виразом природної мови «частина». Мереологічні відношення [16] є транзитивними, але ці відношення поділяються на сім груп, і транзитивність діє лише всередині кожної групи і не прийнята між відношен-

нями різних груп: 1. Компонент-об'єкт; 2. Член-колекція; 3. Частина-об'єкт; 4. Матеріал-об'єкт; 5. Властивість-діяльність; 6. Стадія-процес; 7. Місце-район.

Асоціативні відношення характеризують зв'язки між поняттями, які не є ні еквівалентними, ні ієрархічними, але такі терміни семантично або концептуально пов'язані, і цей зв'язок має бути чітко визначений контрольованим словником. Деякі асоціативні відношення є предметно-специфічними і представляють відношення без їх логічних характеристик (таких як симетричність або транзитивність). Використання явних зв'язків між такими термінами надає додаткові можливості для індексації, пошуку чи обчислення семантичної подібності понять [17], які належать до різних ієрархій або термінологічних систем. Такі відношення можуть пов'язувати ІО різних типів. Приклади таких відношень – "є автором", "має компетенцію", "використовується для створення".

Властивості представлення використовуються для подання знань щодо ПрО, складніших, ніж зв'язки між двома поняттями та визначеннями цих зв'язків. Онтології – це СОЗ, які використовують такі характеристики для явної специфікації концептуалізації домену. Онтології збагачують класифікаційну структуру, яку використовують таксономії та тезауруси: вони не лише відображають відношення між парами об'єктів ПрО, а й можуть містити також непорожні набори правил та аксіом, які визначають вимоги та обмеження щодо використання цих відношень та властивостей. Такий підхід забезпечує підтримку досить складного логічного висновку. Приклад правила – "Якщо об'єкти А та В знаходяться у відношенні Х1, а об'єкти А та С знаходяться у відношенні Х2, тоді об'єкти В та С знаходяться у відношенні Х3 та мають властивість Р зі значенням РРР". Приклад обмеження – "Якщо об'єкти А та В знаходяться у відношенні Х1, а об'єкти А та С знаходяться у відношенні Х2, тоді об'єкти В та С не можуть знаходитися у відношенні Х3 та не мають властивості Р зі значенням РРР".

Постановка задачі

Ціллю даного дослідження є аналіз застосування різних типів СОЗ для організації та вдосконалення бази знань семантизованих Wiki-ресурсів, які містять гетерогенний мультимедійний контент великого обсягу та мають складну структуру, інтегруючу знання із різних ПрО. Актуальність проблеми посилюється через потребу у якісних національних інформаційних ресурсах в умовах гібридної війни, коли визначальними факторами ефективності є як можливість отримання відповідей на складні інформаційні запити, так і достовірність та актуальність отриманої інформації. Це підвищує значення офіційних державних порталів, які мають інтегрувати дані з різних галузей знань та унеможливлювати перекручення (як випадкові, так і зловмисні) інформації у ресурсах з відкритою генерацією контенту.

Wiki-технології та СОЗ

Зараз багато Web-орієнтованих ІР, створених в результаті колективної діяльності користувачів, базуються на технологіях Web 2.0 [18]. Контент таких ІР є більш динамічним та актуальним. Приклад найбільш успішних платформ Web 2.0, які надають механізми підтримки колаборативне створення контенту Web, – Wiki-технології [19], які забезпечують створення структурованих ІР великого обсягу. Однією з поширених реалізацій Wiki-технології є MediaWiki [20]. Існує велика кількість розробок на основі MediaWiki, найбільш відомими з яких є Вікіпедія, Wikibooks, Wiktionary, Wikidata.

Семантичні розширення Wiki-технології спрямовані на додавання змісту елементам Wiki-ресурсу, що робить їх придатними для автоматизованої обробки та аналізу на рівні знань. Воно дозволяє визначати та знаходити інформаційні об'єкти зі складною структурою, що є типовими для певної предметної області. Існує багато підходів до семантизації Wiki-технологій, більшість з яких базується на стандартах проєкту Semantic Web. Для них вже існують

формальні моделі, мови подання, методи обробки та програмні засоби. Одним із них є Semantic MediaWiki – семантичне розширення MediaWiki (www.mediawiki.org/wiki/MediaWiki), яке забезпечує інтелектуальну організацію та пошук контенту ІР [20]. Прикладом складного ІР на основі Semantic MediaWiki є е-ВУЕ [21] – портальна версія Великої української енциклопедії, яка використовує MediaWiki версії 1.34.0 та семантичний плагін Semantic MediaWiki версії 3.1.5.

Semantic MediaWiki забезпечує структуроване подання знань та можливість їх пошуку на змістовному рівні. Але, якщо такі ІР, – приміром, енциклопедії національного рівня – мають великий обсяг та складну структуру, то вони потребують використання сучасних методів менеджменту розподілених знань та систем, що забезпечують ефективну організацію таких знань. У багатьох випадках для цього доцільно застосовувати СОЗ, різні типи яких дозволяють враховувати як специфіку Wiki-технологій, так і особливості організації багатогалузевих енциклопедій та довідників.

Wiki-онтології

Wiki-онтологія є окремим випадком онтології. Вона формалізує знання, представлені в ІР, що розроблений на основі технології Wiki та її семантичних розширень [30]. Виразність онтології Wiki має деякі обмеження, оскільки така онтологія містить лише ті знання, які можна отримати безпосередньо з розмітки Wiki. Наприклад, вона не може визначати характеристики для властивостей об'єкта та властивостей даних, таких як еквівалентність і можливість перетину. В багатьох випадках семантичні розширення Wiki-технологій мають вбудовані засоби для автоматичної або автоматизованої генерації таких онтологій. Скажімо, в технологічному середовищі Semantic MediaWiki онтологія Wiki може бути створена автоматично на основі будь-якої колекції Wiki-сторінок).

З іншого боку, формування Wiki-онтології (або хоча б її структури) може передувати розробці самого Wiki-ресурсу. В цьому випадку певна еталонна онтологія, що створюється експертами та інженерами зі знань, задає базові поняття ПрО та визначає коректні відношення між ними. В процесі розробки ІР за його контентом генеруються поточні Wiki-онтології, які порівнюються з еталонною для того, щоб перевірити правильність подання знань у ресурсі. Чим складніше онтологія ресурсу, тим точніше можна відобразити ПрО, але тим складніше її аналізувати та співставляти з іншими онтологіями. Тому для ефективної розробки семантичного ІР проблема вдалого вибору рівня складності СОЗ є важливим фактором. Від цього залежить і складність Wiki-онтології.

Слід розрізняти Wiki-онтології, які можуть бути згенеровані за звичайними (не семантизованими) ІР та за ІР із семантичною розміткою. Надалі розглядається семантизація Wiki-технології MediaWiki на основі Semantic MediaWiki.

Wiki-онтологія $O_{wiki\_no\_semant}$ для несемантичного ресурсу Wiki містить такі компоненти: $X = X_{cl} \cup X_{ind}$ – це набір понять онтології, де X_{cl} – набір класів, що співпадає з набором категорій Wiki, представлених в обраному наборі сторінок; X_{ind} – це набір екземплярів класів, що створюється як об'єднання імен обраних Wiki-сторінок $P = P_{user} \cup P_{template} \cup P_{spec}$, де P_{user} – набір сторінок, створених користувачами, $P_{template}$ – набір сторінок, що описують шаблони Wiki, P_{spec} – набір інших спеціальних сторінок, які явно відібрані для генерації онтології (як-от, сторінок семантичного пошуку); $R = L \cup \{r_{ier\_cl}\} \cup \{r_{class\_individual}\}$ – набір відношень між елементами онтології, де $L = \{"link"\}$ – набір з одного елемента, який описує посилання з однієї Wiki-сторінки цього ресурсу на іншу; r_{ier_cl} є ієрархічним відношенням між категоріями Wiki-ресурсу, що визначається у процесі створення нових категорій, $r_{class\_individual}$ є ієрархічним відношенням між категоріями та сторінками Wiki-ресурсу, що належать до цих категорій;

$F = \{f_{equ}\}$ – це одноелементна множина, що містить відношення, яке можна використовувати для логічного виведен-

ня в онтології, – відношення еквівалентності між сторінками Wiki, яке пов'язує відсильні Wiki-сторінки. Інші елементи онтологічної моделі цієї Wiki-онтології представлені порожніми множинами.

Формальна модель семантично розмічених Wiki-ресурсів O_{wiki_semant} є складнішою за $O_{wiki\_no\_semant}$ і містить низку елементів, пов'язаних із семантичними властивостями:

$R = L \cup \{r_{ier\_cl}\} \cup \{r_{class\_individual}\} \cup L_{sem\_prop}$, де до R додано набір семантичних властивостей L_{sem_prop} із областю значень у множині Wiki-сторінок;

T – це набір типів даних (наприклад, «текст», «число») для значень властивостей даних.

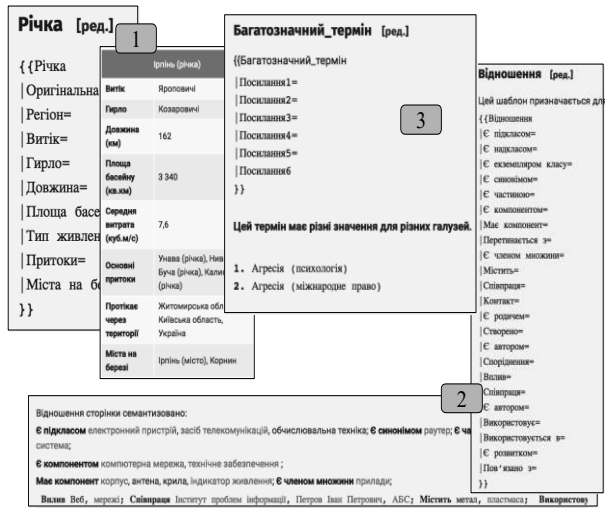


Рис.1. Приклади шаблонів e-BYE для різних груп відношень.

Шаблони Semantic MediaWiki

Можна виділити три групи шаблонів, котрі застосовуються в складних ІР на основі Semantic MediaWiki (наприклад, в e-BYE), які найбільше впливають на складність Wiki-онтології та СОЗ, що використовується для цього: 1. шаблони типових ІО, які дозволяють досліджувати область значень об'єктних властивостей Wiki-онтології і типи відношень між Wiki-сторінками; 2. шаблони для встановлення змісту відношень між довільними Wiki-сторінками, які групують семантичні властивості як за їх логічними характеристиками, так і за сферою використання (це може бути загальний шаблон "Відношення" або відповідний спеціалізований шаблон – "Відношення еквівалентності", "Мереологічні відношення" тощо); 3. шаблони для визначення однотипних відношень (ієрархічних, синонімічних тощо), характерних для найпростіших окремих випадків онтологій (рис.1).

Приклади першої групи шаблонів у e-BYE – "Персоналія" (відношення "Місце народження"), другої – шаблон "Мереологічні відношення" (відношення "Є складовою"), третьої – шаблон "Багатозначний термін" (відношення "Значення1"). У розробці шаблонів для конкретного ІР слід враховувати, як реалізовані в них семантичні відношення вплинуть на рівень складності СОЗ, яка забезпечить керування знаннями.

Висновки

Складність Wiki-онтології визначається тим набором семантичних властивостей типу "сторінка", що використовуються у відповідному Wiki-ресурсі) і, таким чином, шаблонами, що використовуються для введення їх значень. Із цього випливає значущість набору таких властивостей та потреба у його повноті та несуперечності: з одного боку, необхідно створити семантичні властивості для всіх тих відношень ПрО, які має відображати Wiki-ресурс, а з іншого – доцільно зробити цей набір найбільш компактним та зрозумілим для тих розробників ресурсу, що мають застосовувати ці властивості для структурування Wiki-контенту. Для цього пропонується розглядати типи відношень, що підтримуються у різних СОЗ, починаючи від найпростіших. Якщо відношення, що розглядається, є релевантним для контенту ІР, то потрібно визначити його ім'я, описати його характеристики та створити у середовищі Semantic MediaWiki відповідну семантичну властивість типу "сторінка". Крім того, потрібно проаналізувати, для яких категорій сторінок може застосовуватися це відношення, і, якщо ці сторінки належать до одного типового ІО (або до групи типових ІО), то додати відповідну семантичну властивість до шаблонів цих типових ІО. Якщо відношення може застосовуватися досить часто, але його не вдається зв'язати з певними типо-

вими ІО, то відповідну семантичну властивість доцільно додати до одного зі спеціалізованих шаблонів, що створюються саме для встановлення змісту відношень між довільними сторінками.

References

1. Soergel, D. (2009). Knowledge organization systems: overview. www.dsoergel.com/UBLIS514DS-08.2a-1Reading4Soergel-KOSOverview.pdf.
2. Gruber T.R. (1993) A translation approach to portable ontology specifications. Knowl. Acquis., 5(2), 199–220.
3. Baader F., Calvanese D., McGuinness D., Nardi D., Patel-Schneider P. (2003) The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press.
4. Rogushina J., Gladun A., Osadchiy V., Priyma S. (2015) Ontological analysis in the Web. – Melitopol State Pedagogical University Bohdan Khmelnytsky. (in Ukrainian)
5. Navigli R., Velardi P. (2008) From Glossaries to Ontologies: Extracting Semantic Structure from Textual Definitions, Ontology Learning and Population: Bridging the Gap between Text and Knowledge, Series information for Frontiers in Artificial Intelligence and Applications, IOS Press, 71-87.
6. Giunchiglia F., Zaihrayeu I. (2007) Lightweight ontologies. http://eprints.biblio.unitn.it/1289/1/071.pdf.
7. Nikonenko A.A. (2009) Overview of knowledge bases of the ontological type. http://dspace.nbuv.gov.ua/bitstream/handle/123456789/8144/27-Nikonenko.pdf?sequence=1. (in Russian)
8. Rogushina J. (2018) Theoretical bases of application of ontologies for semanticization of Web resources. Problems of programming, (2-3), 197-203. (in Ukrainian)
9. Zaihrayeu I , Sun L., Giunchiglia F., Pan W., Ju Q., Chi M., Huang X. (2007) From web directories to ontologies: Natural language processing challenges. 6th International Semantic Web Conference (ISWC 2007). Springer.
10. Giunchiglia F., Marchese M., Zaihrayeu I. (2006) Encoding Classifications into Lightweight Ontologies. The Semantic Web: Research and Applications, ESWC 2006, 80-94. http://www.science.unitn.it/~marchese/ pdf/P4_eswc06_Encoding.pdf.
11. Rogushina J., Gladun A. (2021) Task Thesaurus as a Tool for Modeling of User Information Needs. In New Perspectives on Enterprise Decision-Making Applying Artificial Intelligence Techniques, , 385-403. Springer, Cham. https://doi.org/10.1007/978-3-030-71115-3_17. https://link.springer.com/chapter/10.1007/978-3-030-71115-3_17.
12. Hjorland B. (2008) What is knowledge organization (KO)? KO Knowledge Organization, 35(2-3), 86-101. https://www.researchgate.net/profile/Birger-Hjorland/publication/277803483_What_is_Knowledge_Organization_KO/links/55d8232608aed6a199a6afce/What-is-Knowledge-Organization-KO.pdf
13. SKOS Simple Knowledge Organization System. (2004). https://www.w3.org/2004/02/skos/.
14. Biagetti M. T. (2020) Ontologies (as knowledge organization systems). ISKO Encyclopedia of Knowledge Organization. https://www.isko.org/cyclo/ontologies.
15. Zeng M. L. (2008)Knowledge organization systems (KOS). Knowledge Organization, 35 (2-3), 160-182.
16. Gladun A., Rogushina J. (2010) Mereological aspects of ontological analysis for thesauri constructing. Buildings and the Environment, 301-308.
17. Rogushina J. (2019) Use of Semantic Similarity Estimates for Unstructured Data Analysis // XIX International Scientific and Practical Conference “Information Technologies and Security” (ITS 2019). CEUR Vol-2577. 246-258. http://ceur-ws.org/Vol-2577/paper20.pdf.
18. Hendler J. A., Golbeck J. (2008) Metcalfe’s law, Web 2.0, and the Semantic Web. Web Sem., 6 (1): 14-20.
19. Wagner C. (2004) Wiki: A technology for conversational knowledge management and group collaboration The Communications of the Association for Information Systems, 13(1), 264-289.
20. Völkel M., Krötzsch M., Vrandecic D., Haller H., Studer R. (2006) Semantic wikipedia. Proc.e of the 15th international conference on World Wide Web, 585-594.
21. Andon P.I., Rogushina J.V., Grishanova I.Y., Reznichenko V.A., Kyrydon A.M., Aristova A.V., Tyschenko A.O. (2021)

Experience of Semantic Technologies Use for Development of Intelligent Web Encyclopedia. UkrPROG, CEUR Workshoop Proc., Vol-2866, 246-259. http://ceur-ws.org/Vol-2866/ceur_246-259andon24.pdf.

22. Rogushina J. (2019) Problems of ontological analysis use for knowledge representation of wiki-resources. Problems in programming, 2, 2019, 17-37. https://doi.org/10.15407/pp2019.02.017. (in Ukrainian).
23. Rogushina J., Gladun A. (2006) Ontologies and multilingual thesauri as a base of semantic retrieval of the Internet resources . Proc. of the XII-th International Conference "Knowledge-Dialogue-Solution", 115-121. (in Russian).

Отримано: 23.03.2022

Про автора:

Рогушина Юлія Віталіївна,
Канд.фіз.-мат.наук, с.н.с Інституту програмних систем НАН України,
публікації в українських виданнях – 200,
публікації в іноземних журналах – 40,
ORCID http://orcid.org/0000-0001-7958-2557.

Місце роботи автора:

Інститут програмних систем
НАН України, 03181, Київ-187,
проспект Академіка Глушкова, 40,
e-mail: ladamandraka2010@gmail.com,
066 550 1999.

УДК 004.94 http://doi.org/10.15407/pp2022.01.34

В.А. Резніченко

60 РОКІВ БАЗАМ ДАНИХ (частина третя)

Наводиться огляд досліджень і розробок баз даних з моменту їх виникнення в 60-х роках минулого століття і по теперішній час. 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 стандартизація, 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 дається короткий огляд досліджень і розробок по базах даних в 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.

Розподілені бази даних.

Розподілена база даних (РоБД) – це інтегрована сукупність баз даних, які фізично розподілені комп'ютерною мережею. А розподілена система управління базами даних (РоСУБД) – це програмна система, яка управляє розподіленою базою даних таким чином, що аспекти розподілення стають прозорими (невидимими) для користувачів. РоСУБД може мати спільний інтерфейс для доступу до розподілених даних [10].

Виникненння РоБД обумовлене тим, що тут природно представляються організаційна структура даних підприємства, підвищується надійність, доступність і локальний контроль, зростає продуктивність, полегшується процедура розширення системи.

Розробка концепції і дослідження РаБД почалися в другій половині 70-х років. Серед численних дослідницьких систем найбільш відомими є три: система SDD-1 [439 - 442], створена в науково-дослідницькому відділенні корпорації Computer Corporation of America наприкінці 1970-х і початку 1980-х років, система System R * [443 - 446], розподілена версія системи -протоколу System R, створена в дослідницькому відділенні компанії IBM на початку 1980-х років, і система Distributed Ingres [447 - 449], розподілена версія прототипу системи Ingres, створена також на початку 1980-х років у Каліфорнійському університеті в Берклі. Варто згадати також проєкт POLIPHEME у Франції [450]. У проєктах 70-х років було виявлено низку ключових проблем, пов'язаних із розробкою систем розподілених баз даних, подані підходи до їх вирішення. Той факт, що всього лише за кілька років у цій царині були отримані значні результати, підтверджується появою наприкінці 70-х оглядів на цю тему [451 - 453].

До кінця 80-х років були здійснені численні дослідження, експериментальні розробки, почали з'являтися перші промислові РоБД. Було звернуто увагу на створення мультибаз даних і на надання більшої автономності індивідуальним системам [454 - 455].

У 1986 – 87 роках були представлені перші промислові РоСУБД Ingres/STAR,

© В.А. Резніченко, 2022

Oracle 7 s DB2. Тож постала необхідність формування основних принципів, вимог і функціональних можливостей РоБД. Відповіддю на ці вимоги в 90-му році стала стаття Дейта [456], де були сформульовані 12 правил РоБД, головне з них – прозорість для користувачів розподіленої структури баз даних.Ці правила були прийняті науковим товариством і ними досі користуються у розробці РоСУБД.

Типи РоБД. Існують два основні типи РоБД: однорідні (homogeneous) і неоднорідні (heterogeneous).

Однорідні РоБД. У них усі вузли перебувають під управлінням РоСУБД одного типу (і, можливо, під управлінням однієї операційної системи). Існують два типи однорідних РоБД: автономні і неавтономні. Автономні працюють незалежно, передаючи і приймаючи повідомлення одне одному для спільного оновлення даних. Неавтономні РоБД передбачають існування центральної (головної) РоСУБД, котра координує доступ до даних і їх оновлення в мережі. Звичайні розподілені (regular distributed) й паралельні бази даних належать до однорідних РоБД.

Неоднорідні РоБД. Вони працюють під управлінням різних операційних систем і типів РоСУБД. Існує чотири типи неоднорідних РоБД:

- федеративні (federated);
- із посередниками (mediators);
- мультибази даних (multidatabases);
- однорангові бази даних.

Федеративні РоБД. Являють собою об′єднання БД різних типів, якими володіють різні користувачі і які об′єднуються для спрощення спільного використання даних. Федеративна БД передбачає визначення глобальної інтеграційної схеми, що містить відображення в схеми баз даних учасників. Уперше федеративну БД визначили Маклеод і Хаймбігнер (McLeod, Heimbigner) 1985 року [457], вона досліджувалася у багатьох працях [458, 459]. У [460] наводиться огляд федеративних БД.

У разі суттєвого збільшення баз даних, що інтегруються, буває важко, а інколи і неможливо визначити глобальну інтеграційну схему. Мультибази даних не передбачають існування глобальної схеми. Натомість мова запитів дає можливість специфікувати вирази, які дозволяють здійснювати пошук за об′єднувальними базами даних.

Посередники (mediators) [462, 463]. Вони розміщені між системами з однією глобальною схемою і взагалі без схем. Натомість користувачі визначають погляди – посередники, які об′єднують і узгоджують дані з різних джерел. Для таких поглядів необхідна мова запитів, яка здатна формулювати запити за багатьма базами даних, подібно до мови запитів мультибаз даних.

Однорангові БД (peer – to – peer databases – P2PDB) [464 - 466]. Вони являють собою сукупність автономних локальних репозиторіїв/баз даних, які взаємодіють між собою на рівноправній основі. Основне завдання P2PDB – розповсюджувати запити між гетерогенними вузлами у великій розподіленій мережі. Таке розповсюдження може зупинитися за кілька кроків. Це допустимо для деяких сучасних систем, які не потребують високої точності результатів. Як – от у пошукових машинах Інтернету.

Розподіл даних. Фрагментація. В РоБД існує задача розподілу логічно цілісної БД по вузлах розподіленої структури таким чином, щоб оптимізувати цільову функцію. Є два фундаментальні методи вирішення цієї задачі: фрагментація і реплікація.

Фрагментація (сегментація, декомпозиція) передбачає розподіл даних на сегменти (фрагменти) даних, що не перетинаються, для їх прив′язування до вузлів мережі. Реплікація передбачає запам′ятовування на різних вузлах ідентичних копій всієї або ж частини логічної бази даних. РоСУБД гарантує користувачам прозорість такого розподілу. Крім цього, існує задача розміщення фрагментованих/реплікованих даних у вузлах мережі.

Є два види фрагментації: горизонтальна і вертикальна. Під час горизонтальної фрагментації відношення розбивається на групи рядків, які розподіляються по вузлах. За вертикальної фрагментації відно-

шення розбиваються на групи стовпчиків. Допускається також гібридна фрагментація, яка передбачає одночасне використання двох попередніх фрагментацій.

Основні дослідження з фрагментації були здійснені на початку 80-х років [467, 469, 470, 10]. Одна з основних задач вертикальної фрагментації – визначення наборів атрибутів, що мають бути об′єднані в одну групу. В працях [471, 472] було запропоновано алгоритм енергетичного зв′язування (bond energy algorithm) для групування атрибутів. На його основі здійснюється вертикальна фрагментація. У праці [473] висунуто модифікований варіант цього алгоритму.

Що ж до задачі розміщення даних, то роботи в цьому напрямку почалися ще наприкінці 60-х років, коли досліджувалась проблема розміщення файлів [474]. У працях [475 - 477] була досліджена проблема складності задач розміщення. В [478, 479] досліджені динамічні алгоритми розміщення даних, які передбачають можливість зміни початкового розміщення для обліку змін у методах доступу й робочих навантаженнях. Пропонувались також методи інтеграції фрагментації та розміщення [467, 468].

Розподіл даних. Реплікація. Роботи із реплікації БД датуються початком 80-х років, коли були здійснені дослідження щодо доступності даних, а більшість пропонованих рішень забезпечувала узгодженість даних. Цікавим оглядом досліджень того часу є стаття [480].

Основна проблема реплікації даних полягає в тому, що оновлення будь – якого заданого логічного об′єкту має розповсюджуватися по всіх збережених копіях цього об′єкту. 1996 року Грей продовжив дослідження в цій царині [481] й запропонував варіанти негайного (eager) і відтермінованого (lazy) оновлень. Один із варіантів відтермінованого оновлення – використання первинної копії (master copy), коли основна копія оновлюється оперативно, а оновлення вторинних копій відкладається до зручного часу. Причому синхронна реплікація передбачає завершення розповсюдження змін до завершення транзакції, а синхронна реплікація допускає розповсюдження змін після завершення транзакції. Зрештою Грей запропонував дворівневе оновлення транзакцій.

Остання стаття активізувала подальші дослідження з реплікації. Серед напрямків досліджень – зменшення накладних витрат на комунікацію і координацію за рахунок затримки оновлень видалених копій. Однак у цьому випадку копії можуть містити застарілі або навіть неузгоджені дані. Через це з′явилися пропозиції щодо неузгодженості [482], встановлення обмежень на «застарілі» дані та виявлення й усунення неузгодженості [483].

Інший напрямок досліджень, які здійснювалися в контексті масштабованої кластерної реплікації, пов’язаний із розробкою методів забезпечення надійної узгодженості за прийнятних витрат [484, 485]. Із появою хмарних систем зберігання в дослідження були залучені внутрішньохмарні реплікації, концептуально схожі на кластерну реплікацію, а також міжхмарні й геореплікації [486, 487]. Що ж до розподілених систем, то спочатку дослідження концентрувалися на реплікаційних файлових системах [488, 489], згодом на веб-серверних реплікаціях [490] і файлових реплікаціях однорангових систем [491]. Також були отримані результати із відмовостійких реплікацій об′єктів [492, 493]. Як і в сфері баз даних, найостанніші результати щодо реплікації розподілених систем, належать до хмарної інфраструктури, а саме – реплікації в системах зберігання [494], таких, як HDFS і Cassandra, а також глобальна (wide – area) реплікація [484, 495].

Тупики в РоБД. У БД, що послуговуються протоколом блокувань для доступу до спільно використовуваних даних, можливі тупикові ситуації (deadlock), коли транзакція очікує події, яка може статися через наступні дії самої транзакції. Як–от, коли дві транзакції чекають одна на одну.

Існують такі категорії алгоритмів виявлення тупиків у РоБД [10]: централізовані, ієрархічні, розподілені. Централізовані алгоритми [496, 497] використовують центральний вузол для виявлення тупиків. Як зазначається в [439], цен-

тралізоване двофазне блокування (two – phase locking – 2PL) і виявлення тупиків стало позитивною природною комбінацією. Централізоване виявлення тупиків уперше було реалізовано в Distributed INGRES [498]. Ієрархічні алгоритми [497, 499] для виявлення тупиків покладаються на ієрархічну структуру вузлів РоБД. Розподілені алгоритми [499, 500] покладаються на кооперацію всіх вузлів РоБД для виявлення тупиків. Розподілене виявлення тупиків уперше було реалізовано в System R* [500].

У [10] представлено огляд методів управління розподіленими тупиками. В оглядах [501 - 504] обговорюються різні розподілені алгоритми виявлення тупиків. У праці [505] наводиться порівняльний аналіз додаткових алгоритмів виявлення тупиків : просування шляху (path – pushing) [499], зондовий (probe – based) [506], глобального стану (global state) [507, 508].

Розподілена обробка запитів. Це процедура виконання запиту в розподіленому середовищі, де дані розміщені в різних вузлах комп′ютерної мережі. Вона передбачає перетворення запиту, сформульованого високорівневою мовою (наприклад, SQL), у вираз процедурної мови низького рівня (приміром, реляційна алгебра), який названо «план виконання запиту». Далі цей план оптимізується з урахуванням розподіленості даних і, зрештою, відбувається послідовне виконання операторами отриманого оптимального плану.

Дослідження із розподіленої обробки запитів почалися наприкінці 70-х років. Тоді було розроблено три експериментальні системи, де були закладені фундаментальні методи розподіленої оптимізації і обробки запитів: SDD-1 [509] (1976), Distributed INGRES [510, 511] (1977) і Sysnem R* [512, 513] (1981). Вважається, що першим дистрибутивним алгоритмом оптимізації запитів є «скелелазіння» (hill climbing) Вонга [514], який згодом було поліпшено в SDD-1 через включення операції напівз′єднання. Оптимізаційний алгоритм SDD-1 є статичним, він спрямований на зменшення сумарних комунікаційних витрат і не підтримує фрагментацію і реплікацію. Дистрибутивний алгоритм оптимізації запитів Distributed Ingres [510] у кожному наступному кроці детерміновано аналізує простір імовірних планів і робить висновок щодо локальної оптимізації. Він також підтримує горизонтальну фрагментацію. Цільова функція оптимізації є виваженою комбінацією вартості сумарного часу і часу реакції. Алгоритм є динамічним.

Дистрибутивний алгоритм оптимізації запитів System R* [513] всебічно аналізує межі пошуку всіх можливих планів виконання запитів. Алгоритм не підтримує фрагментацію і реплікацію. Цільова функція оптимізації враховує локальну обробку й комунікаційні витрати. Алгоритм є статичним.

Було здійснено дослідження з оптимізації виконання виражень реляційної алгебри із розподіленим середовищем включно. В статті [515] наведено огляд цих результатів. Запропоновано кілька підходів щодо динамічної оптимізації запитів для паралельних і розподілених баз даних [516]. Алгоритм у [517] передбачає зміну плану відпрацювання запиту в процесі його виконання, аби врахувати непередбачувані обставини. В системі Mariposa [518] вперше була висунута модель оптимізації розподіленого запиту. В монографіях [519, 10] детально висвітлюються результати в галузі технологій розподілених баз даних і оптимізації розподілених запитів, отримані у 80 – 90-х роках. Стаття [516] є найновішим оглядом у цій сфері.

Управління паралелізмом (Concurrency Control). Це процедура такого управління одночасною безконфліктною роботою багатьох транзакцій, за якої транзакції коректно виконують свою роботу без порушення обмежень цілісності БД (принципа ACID).

Дослідження з управління паралелізмом у розподілених системах зародилися на початку 80-х років. Вони спиралися на широко відому на той час статтю [520] із управління паралелізмом у централізованих БД. Грей розвинув ці ідеї для транзакцій [521], а Спектор і Шварц [522] дослідили транзакції у розподіленому середовищі.

Було запропоновано три механізми управління паралелізмом: блокування, оптимістичний протокол і впорядкування за часовими позначками (timestamp odering).

Блокування. Це обмеження доступу до спільно використовуваних ресурсів (даних) із одночасним виконанням багатьох транзакцій.

Першим широко відомим механізмом блокувань було двофазне блокування (Two – Phase Locking – 2PL), яке вперше описане в [520]. Згодом була визначена велика кількість його різновидів: строга (strict), консервативна (conservative), первинної копії (primary copy), розподілена (distributed), точна (rigorous). У [523] описано варіант 2PL з урахуванням використання старих значень. Також були запропоновані гібридні блокування, що передбачають використання методів, відмінних від 2PL [524 - 526].

Наступний тип управління паралелізмом дістав назву «оптимістичний» у тому сенсі, що створюються локальні копії даних транзакції й оновлюються саме вони, а не власне дані. Вперше цей метод був висунутий у праці [527], і відтоді досліджено чимало його різновидів [528, 529].

Зрештою впорядкування за часовими позначками використовує Системний Час або ж певний логічний лічильник в ролі часових позначок для впорядкування виконання паралельних транзакцій. Транзакції присвоюється часова позначка здебільшого із урахуванням часу запуску транзакції. Старіша транзакція є пріоритетнішою. У разі конфлікту перевага надається пріоритетній транзакції. Цей протокол описано в [530 - 532]. У [532, 533] також описані багатоверсійні часові позначки. Цікавий огляд методів управління паралелізмом знаходимо в статті [534].

Машини баз даних.

Загалом машиною бази даних (МБД) прийнято називати апаратно-програмний мультимікропроцесорний комплекс, призначений для всіх або деяких функцій СУБД. Цей напрямок баз даних з'явився на початку 70-х років. На першому етапі протягом 10 – 12 років основна ідея досліджень і розробок МБД була спрямована на створення спеціальних обчислювальних обладнань і розробку архітектур, де процес обчислення бази даних розміщувався ближче до дисків з метою значного збільшення продуктивності. На той час було реалізовано понад 50 проєктів. Основними критеріями оцінки того чи іншого проєкту були повнота виконуваних функцій СУБД і очікуване підвищення продуктивності. На основі експериментальних прототипів у багатьох країнах світу згодом сформувалося виробництво різних зразків машин баз даних [535].

У цей період були запропоновані рішення, які дістали назву процесорів фільтрів. Їхньою задачею була перевірка передаваних даних із дисків на зовнішній сервер. У працях [536, 537] процесори фільтрів були розділені на такі групи:

процесор на доріжку (Processor-per-Track - PPT),

процесор на голівку (Processor-per-Head - PPH),

процесор на диск (Processor-per-Disk – PPD),

мультипроцесорний кеш (Multi-Processor Cashe – MPC),

процесор на комірку бульбашкової пам'яті (Processor-per-Bubble-cell – PPB).

Процесор на доріжку – PPT.

Згідно із [538] першим дослідником в галузі МБД був Даніель Слотник (Daniel L. Slotnick), який 1970 року опублікував статтю [539], де висунув пропозицію архітектури з процесором на кожну доріжку. В цій архітектурі запам'ятовувач складається з великої кількості комірок, кожна з яких має доріжку даних, зв'язану з процесором, котрий блискавично виконує функцію пошуку потрібних даних. Координацію роботи з комірками здійснює управляючий процесор. Основна ідея Слотника полягала в тому, щоби здійснювати пошук у базі даних безпосередньо на запам'ятовувачі. Тим самим обмежити обсяг даних, які передаються на основний процесор. У подальшому підхід Слотника розвинули Паркер (Parker) [540], Мінскі (Minsky) [541] і Пархамі (Parhami) [542]. На основі цієї архітектури реалізовані МБД RAP [543], CASSM [544], RARES [545].

Даніель Слотник

Процесор на голівку – PPH. До цього класу належать МБД, в яких логіка обробки даних прив′язується до кожної голівки в диску з рухомими голівками. В PPH дані паралельно передаються від головок до багатьох процесорів. Кожен процесор застосовує функцію відбору до вихідного потоку даних і розміщує вибрані дані у вихідному буфері. За такої організації кожен циліндр диску з рухомою голівкою аналізується щокожен оберт. До цього класу належать МБД DBC [546], SURE [547].

Процесор на диск – PPD. На відміну від PPT і PPH дана архітектура передбачає використання стандартних дисководів. Процесор (або багато процесорів) розміщується між диском і запам′ятовувачем, куди передаються відібрані дані. Цей процесор діє як фільтр, передаючи до основного процесора лише ті дані, які відповідають критерію відбору.

Мультипроцесорний кеш – MPC. До цього класу належать МБД, в яких спеціалізовані процесори відділяються від пристроїв зберігання великим дисковим кешем. Мета цього архітектурного вирішення – підтримувати паралелізм обробки під час використання традиційних пристроїв зберігання. Перед обробкою дані мають бути переміщені з диска в кеш, після чого вони стають доступними процесорам у паралельному режимі. Більше того, проміжні результати виконання запиту поміщаються процесорами в кеш і до них надається швидкий доступ для виконання наступних операцій запиту. Реалізовано багато МБД цього класу, як от RAP2 [548], DIRECT [549], INFOPLEX [550], RDBM [551], DBMAC [552].

Процесор на комірку бульбашкової пам′яті – PPB. З кожною коміркою зовнішньої пам′яті асоціюється процесор.

Варто зауважити, що в цей період більшість проєктів розробки МБД концентрувалася навколо спеціалізованого апаратного забезпечення, яке перебувало ще в стадії розробки. Зокрема, такого, як CCD-пам′ять (charge-coupled device, пристрій із зарядовим зв′язком), бульбашкова пам′ять (bubble memory), диски з фіксованими голівками на кожну доріжку (hesd-per-track disks), оптичні диски (optical disks). Жодна з цих технологій уповні не виправдала себе. Тож після дванадцяти років активності в цьому напрямку майбутнє МБД виглядало непевно навіть для найбільших її прихильників. Так, наприклад, 1983 року стаття [538] передрікала стрімке зникнення МБД. Найвідомішими монографіями на тему машин баз даних першого етапу були праці Есена Озкарахана (1986р.), а також Калиниченка Л.А. і Ривкіна В.М. (1990р.) [9].

Есен Озкарахан

Попри песимістичні настрої, напрямок МБД вижив і успішно розвивається завдяки паралельним системам баз даних.

Як зазначається в [553], успіх паралельних баз даних пояснюється широким розповсюдженням реляційних баз даних. 1983 року вони лише почали з′являтися на ринку, сьогодні ж – домінують. Реляційні запити щонайкраще підходять для паралельного виконання; вони складаються

з однорідних операцій над однорідним потоком даних. Кожна операція утворює нове відношення, тож із операцій можуть складатися високо паралельні графи потоків даних. Дві операції можуть працювати послідовно, якщо направити вивід однієї операції на вхід іншої. Це так званий конвеєрний паралелізм (pipelined parallelism). Якщо розділяти введені дані між кількома процесорами і пам'яттю, часто-густо з'являється можливість розбити операцію на кілька незалежних операцій, кожна з яких працює з частиною даних. Такий розподіл даних і обробка мають назву «роздільний паралелізм» (partitioned parallelism).

Таким чином, історія демонструє, що вузькоспеціалізовані машини баз даних виявилися неспроможними, тоді як паралельні системи баз даних досягли величезних успіхів. Успішні паралельні системи баз даних базуються на звичайних процесорах, пам'яті і дисках. Саме в цих системах здебільшого відобразилися ідеї високо паралельних архітектур.

У 1980-х роках дослідження щодо машин баз даних були зосереджені на масивних паралельних обчисленнях (massive parallel computing). Процесори були з'єднані у вузли, і такі вузли потім об'єднувалися у високошвидкісні міжблокові зв'язки [553, 554]. Деякі з цих типів машин баз даних досягли значного успіху в промисловості.

У середині 80-х років Стоунбрейкер висунув наступну просту класифікацію паралельних мультипроцесорних систем [555].

Спільно використовувані пам'ять і диски (shared-everything – SE). Всі процесори мають прямий доступ до загальної глобальної пам'яті і до всіх дисків. Взаємодія між процесорами відбувається з використанням загальної пам'яті. Прикладами таких систем можуть бути XPRS [556], DBS3 [557]? Volcano [558].

Спільно використовувані диски (shared disks – SD). Кожен процесор має власну пам'ять і прямий доступ до всіх дисків. Усі процесори пов'язані один з одним через високошвидкісну мережу для передачі даних. Прикладами паралельних систем баз даних SD-архітектури є IBM IMS [559], Oracle Parallel Server [560], nCUBE [561], VAXclusters [562], IBM Parallel Sysplex [563].

Відсутність спільного використання ресурсів (shared-nothing – SN). Кожна пам'ять і диск є у розпорядженні одного процесора, який працює як сервер збережених в них даних. Масовий запам'ятовуючий пристрій у таких архітектурах розподілений між процесорами через з'єднання одного чи більше дисків. Так само, як і в SD-архітектурі, всі процесори зв'язані один з одним через високошвидкісну мережу. Відсутність спільного використання ресурсів характерна для систем баз даних, які використані в проєктах Teradata[564], Gamma [565], Tandem [566], Bubba [567], Arbre [568], і nCUBE [569]. Прикладами комерційних систем SD-архітектури є NonStop SQL [570], Informix PDQ [571], NCR/Teradata DBC [572], IBM DB2 PE [573].

Аналізу архітектур паралельних систем баз даних присвячена також стаття Соколинського Л.Б. [574].

Згодом з'явилися мультипроцесорні системи, які поєднували характеристики SE- і SN-архітектур, тому Коупленд і Келлер [575] запропонували розширити класифікацію Стоунбрейкера наступним чином:

- кластеризовано все (clustered everything – CE) – кластери з SE-архітектурою об'єднуються за принципом SN-архітектури;

- кластеризовані диски (clustered-disk – CD) – кластери з CD-архітектурою об'єднуються за принципом CN-архітектури.

Такі архітектури дістали назву ієрархічних [576]. Пропозиції Коупленда дозволяють створювати дворівневі ієрархії (ISE/SD-кластери першого рівня з'єднуються в SN-кластери другого рівня). Дворівнева архітектура Коупленда може бути легко розширена до архітектур із трьома чи більше ієрархічними рівнями. Дворівнева ієрархічна архітектура була досліджена в працях [575, 577 - 580].

Зазначимо, що в другій половині 90-х років з'явилися багатопроцесорні системи із компонентами складної конструкції. Вони увібрали в себе різні архітектурні

рішення, що не підпадають під класифікацію Стоунбрейкера і Коупленда. До них можна віднести мультипроцесорну систему серії МВС-100/1000 [581], мультипроцесорну систему SP-2 [582] компанії IBM, комп'ютери на основі технології Server Net компанії Tandem [583], гібридну архітектуру CDN [584].

За [585] першим кроком на шляху створення сучасних МБД була презентація 2000 року технології InfiniBand – високошвидкісної комутованої комп'ютерної мережі компанії Voltaire (партнер Oracle, починаючи з 2001 року), яка була використана в Oracle RAC (Real Application Cluster), починаючи з версії Oracle Database 9i. 2009 року серед Top 500 суперкомп'ютерів світу 29% використовували InfiniBand. Oracle Exadata VI була першою сучасною MBD, створеною Oracle HP (Hewlett-Packard) 2008 року. Тестування цієї MBD у CERN [586] показало високу ефективність за часом і пам'яттю в процесі розвантаження даних великого обсягу. Sun і Oracle створили MBD Exadata Database Mashine Version 2. Завдяки застосуванню сучасних технологій цих двох компаній MBD працює вдвічі ефективніше за Oracle Exadata VI.

Teradata Database – це система масової паралельної обробки (MPP) із колективною розподіленою архітектурою. Задача рівномірно розподіляється по всіх процесах і паралельно обробляється. Підтримує архітектуру без спільного використання ресурсів. Має високу горизонтальну масштабованість, а також один із найрозвинутіших оптимізаторів на ринку. Автоматично рівномірно розподіляє дані по дисках. Підтримує стандарт SQL.

Бази даних, що підтримують роботу з масивами. БД масивів дає змогу представляти і маніпулювати багатомірними масивами однорідних даних. Вважається, що попередником БД масивів була створена 1982 року PICDMS [587] – СУБД для роботи з рисунками. Вона уможливлює оперування двомірними масивами з допомогою процедурної мови.

1993 року Майєр і Венс [588] констатували, що технологія баз даних вкрай рідко використовується в наукових додатках тому, що СУБД не підтримують структури з упорядкованими даними, зокрема, такими, як масиви. Ця заява співпала з початком активного розвитку досліджень і розробок із БД масивів.

Значний внесок у розвиток теорії і практики СУБД масивів зробив німецький учений Пітер Бауманн (Peter Baumann). Він перший 1994 року запропонував декларативну мову запитів для роботи з багатомірними масивами, які базуються на пропонованій ним же алгебрі багатомірних масивів [589, 590]. Розроблені алгебра і мова запитів стали основою створення 1996 року під його керівництвом першої СУБД масивів RasDaMan [591], яка підтримувала реляційну модель даних із додатковим типом даних «багатомірний масив» і спеціальною мовою запитів RASQL, що базується на SQL. Згідно з даними [592] обсяг даних на всіх пристроях RasDaMan наближається до петабайту.

Пітер Бауманн

Моделі і мови. Було запропоновано численні формальні моделі й мови баз даних масивів, аналіз яких можна знайти в [593, 594]. Наведемо деякі.

Алгебра карт (Map algebra) [595, 596] – алгебра, що базується на множинах, розроблена на початку 80-х років Даною Томлін (Dana Tomlin) для маніпулювання географічними даними. Представляє двомірні і тримірні растрові дані. В ній відбувається категоризація операцій над масивами залежно від кількості комірок вхідного масиву, які беруть участь у створенні комірки вихідного масиву.

AFATL Image Algebra [597] – це алгебра, розроблена для обробки зображень і отримання статистичної інформації.

AML (Array Manipulation Lanquage) [598] – універсальна мова маніпулювання масивами, що базується на пропонованій авторами алгебрі багатовимірних масивів. Характерною рисою AML є поняття бітових шаблонів і шаблонно-орієнтованих функцій.

AQL (Array Query Language) [599, 600] – ця мова вмонтовує підтримку багатовимірних масивів у мову NCRA, яка є розширенням мови вкладеного реляційного обчислення NRC.

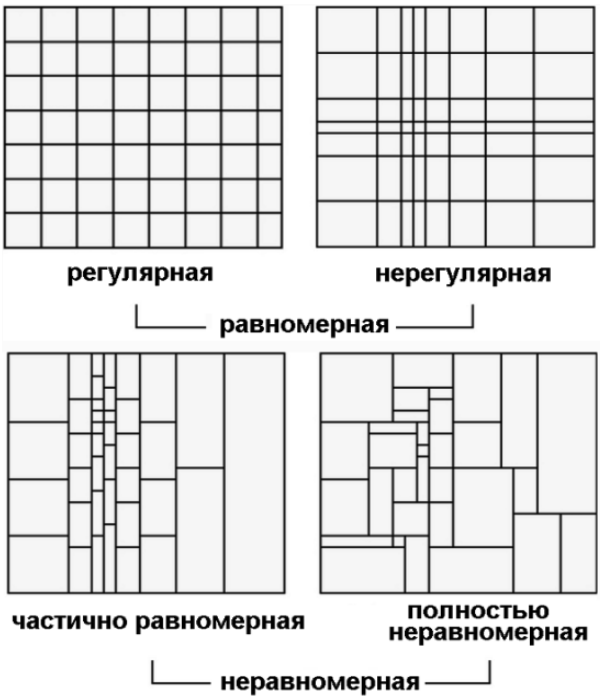
Array Algebra [589, 590] – пропонується алгебраїчна модель масиву, яка базується на трьох ортогональних примітивах, щодо яких надається набір допоміжних функцій. Цей набір обумовлюється використовуваною моделлю даних (об′єктивною або реляційною).

RAM [601, 602] – модель розроблена як розширення реляційної СУБД MonetDB [603].

Зберігання масивів. Зазвичай великі багатомірні масиви розбиваються на підмасиви, що утворюють одиниці доступу до них. Таке розбиття дістало назву мозаїки (tiling), а елементи мозаїки – плитки (tile) [618]. Мозаїка складається з плиток, що не перетинаються, кожна плитка – багатомірний підмасив висхідного масиву. Виділяють два основні види мозаїки – рівномірна (aligned) і нерівномірна (nonaligned). Рівномірна мозаїка для n-мірного масиву означає, що вона формується гіперплощинами, які є ортогональні вісям n-мірного простору і розбивають весь масив на «плитки». Якщо всі площини знаходяться на одній відстані, то така мозаїка називається регулярною рівномірною, в іншому випадку – нерегулярною. В нерівномірній мозаїці (nonaligned tiling) деякі плитки мають сторони, які не є продовженням сторін сусідніх плиток. У частково рівномірній (partially aligned) мозаїці плитки вирівняні принаймні по одному з вимірів, а в повністю нерівномірній (totally nonaligned) таких вимірів немає. На рисунку, взятому з [618], наводиться приклад графічної інтерпретації цих чотирьох категорій мозаїки для двомірного простору.

Архітектура реалізації. Виділяються наступні варіанти архітектури реалізації систем БД масивів:

- повнофункціональні системи БД масивів, реалізовані з нуля (RasDaMan [591], SciDB [604], MonetDB/SciQL [605];
- реалізовані у вигляді додаткових рівнів у існуючих СУБД (EXTASICID [606, 607];
- реалізовані у вигляді об′єктно-реляційних розширень (PostGIS Raster [608], Teradata Arrays [609], Oracle GeoRaster [610].



Пропонувалося два способи «впровадження» масивів у реляційні БД:

- додавання масивів у вигляді нового типу стовпчика (Ras DaMan, Teradata, Oracle, PostGIS, Roster, ISOSQL);
- подання масивів у вигляді таблиці (SciQL і SciDB).

2007 року на симпозіумі з екстремально великих баз даних (XLDB) представники науки і промисловості дійшли висновку, що існуючі СУБД не спроможні маніпулювати обсягами даних, які з′являться в найближчому майбутньому. Була також підкреслена необхідність розробки СУБД нового покоління, що мають відповідати таким вимогам [611]:

- модель даних базується на багатовимірних масивах, а не на кортежах;
- модель зберігання базується на версійності, а не на оновленні значень;
- масштабованість до сотень петабайт і висока відмовостійкість;
- СУБД є вільно розповсюджуваним програмним забезпеченням.

Відповіддю на це звернення став запуск 2008 року міжнародного проєкту під керівництвом Майкла Стоунбрейкера із створення нової СУБД під назвою SciDB. 2010 року було випущено першу публічну версію SciDB [612]. Її архітектура заснована на модифікованому ядрі Postgress. SciDB призначена для зберігання, обробки й аналізу надвеликих обсягів багатовимірних розподілених масивів наукових даних, масштабованих на тисячі серверів [613]. Зберігання даних організовано у вигляді багатовимірних вкладених масивів, для обробки яких розроблені мови AQL (Array Query Language) і AFL (Array Functional Language).

Інші системи БД масивів.

SciQL [605]. Мова запитів, що заснована на SQL і використовує масиви для наукових застосувань. Розширює колончату СУБД MonetDB операторами над масивами [614, 615], тим самим дозволяючи MonetDB ефективно функціонувати як база даних масивів.

EXTASCID [606, 607]. Це повна і розширювана система для обробки наукових даних. Підтримує як масиви, так і реляційні дані. Створена на основі масивно-паралельної архітектури GLADE для агрегування даних.

PostGIS Raster [608] (раніше відома як WKT Raster) дає змогу підтримувати растрові дані в системі PostGIS. Це забезпечується визначенням нового типу даних RASTER і додаткового набору SQL-функцій, котрі працюють із векторними і растровими даними.

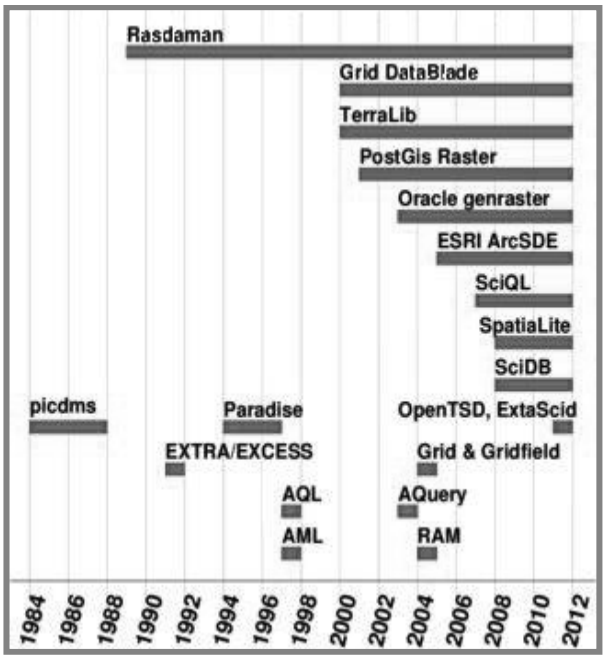
Oracle GeoRaster [610] – це вбудована в Oracle Spatial можливість збереження, індексування, аналізу й доставки растрових зображень (як-от супутникових знімків), даних типу grid-даних, а також зв'язаних з ними метаданих. Ці типи даних можна використовувати для зберігання багатовимірних grid-шарів і електронних зображень, які можуть бути прив′язані для позиціювання на поверхні Землі або в локальній системі координат.

Teradata Arrays [609]. Нещодавно Teradata ввела в свою СУБД масиви у вигляді самостійного типу даних.

Для ілюстрації історії розвитку систем БД масивів наведемо рисунок із [616].

У 2018 році до ISO SQL було включено підтримку багатовимірних масивів даних [617] у вигляді спеціального типу даних.

На завершення зазначимо, що Альянс із Дослідницькими Даними (RDA – Research Data Alliance) надав 2021 року вичерпний огляд баз даних масивів і пов′язаних з ними Технологій [594].



Статистичні бази даних.

Під статистичними базами даних (СБД) маються на увазі такі БД, які дають можливість отримувати, зберігати і обробляти агреговані дані, тобто дані, отримані з допомогою різних способів узагальнення, групування, класифікації.

Дослідження статистичних баз даних почалися у 1970-х роках, а найбільшого розвитку набули в 1980-х роках ще до появи OLAP і їхній розвиток триває й дотепер.

Статистичні моделі даних.

Статистичні дані абстрактніші за звичайні, а операції мають іншу семанти-

ку. В СБД аналіз здійснюється з використанням агрегованих даних, отриманих із необроблених. Зведені дані мають бути різних форм, які не підтримуються традиційними СУБД. Більше того, реляційна модель у чистому вигляді теж не підходить для обробки таких даних. Основна причина – багатовимірність статистичних даних. Тож для СУБД потрібні або нові структури даних і операції над ними, або ж необхідно розширювати реляційну модель даних, аби мати можливість представляти відношення над множинами й нові оператори до них [630]. У СБД визначено три типии моделей даних: графічні, табличні й багатовимірні. Подамо короткий опис статистичних моделей даних, про які докладну інформацію знаходимо в працях [619, 620]:

SUBJECT [621] – подана графічна модель системи SUBJECT;

SAM (Semantic Association Model) [622] – модель було розроблено для моделювання як наукових статистичних даних, так і бізнесорієнтованих даних;

GRASS (Graphical Approach for Statistical Summaries) [623] – є розширенням SUBJECT. Для представлення моделі використовується орієнтований, направлений, ациклічний граф;

CSM (Conceptual Statistical Model) [624] – використовуються дві різні, однак доповнюючи одна одну моделі даних для опису елементарних і зведених даних, а саме ER-модель Чена і перевизначену модель GRASS;

STORM (Statistical Object Representation Model) [625] – графічна модель, в якій логічне представлення відокремлене від фізичної структури статистичних таблиць;

MEFISTO [626] – функціональна модель даних, що базується на структурі з назвою «статистична сутність», а також на численних операціях, які складають алгебру маніпулювання даними цієї структури;

Розширена реляційна модель даних із включенням у реляційну алгебру додаткових статистичних операторів [627];

Темпоральна статистична модель даних [628].

Статистичні оператори (алгебри). У статтях про СБД пропонується чимало підходів щодо визначення операторів, які б відповідали обраній структурі моделі. В праці [629] вводяться статистичні оператори, аналогічні реляційній алгебрі, але із семантикою, характерною для багатовимірних об′єктів. У подальшому вводиться поняття повноти за аналогією з реляційною повнотою і показується повнота запропонованої алгебри. В праці [630] пропонується також розширення реляційної моделі даних шляхом введення відношень над множинами й оператори до них. Ще одним прикладом алгебри, залежної від обраної статистичної моделі, є праця [631], де задіяна двомірна модель презентації статистичних даних. 1997 року в [632] було висунуто варіант розширення SQL функціональними можливостями OLAP для отримання підсумкових значень у багатовимірному просторі. У працях [633, 634] пропонуються багатовимірні моделі даних і оператори. Статті [626, 635] також присвячені операторам у СБД. У [636] представлена алгебра статистичних даних.

Метадані. Вважається, що статистичні дані мають два типи атрибутів [637]: вільні атрибути, що являють собою результати застосування до висхідних даних агрегуючих функцій, і дескриптивні атрибути, які ці вільні дані описують, що також називаються метаданими. Правильно організовані, класифіковані й описані метадані дуже корисні для розуміння суті зведених даних. У зв′язку з цим ефективне використання метаданих є вкрай важливим у СБД. Додаткову інформацію про це можна отримати в працях [638 - 642].

Системи й мови запитів статистичних баз даних. В огляді [643] дано всебічний аналіз систем і мов запитів статистичних баз даних на основі таксономії, запропонованій в [644]. Перерахуємо їх, посилаючи зацікавлених читачів до вказаних статей для детального ознайомлення.

1)Статистичні системи управління базами даних (ССУБД), створені на основі традиційних СУБД. Більшість ССУБД даної категорії створені на основі реляційних СУБД. До них належать:

STRAND [645] базується на ER-моделі Чена, є похідним від CABLE [646]

і має за основу реляційну СУБД INGRES. Запити STRAND транслюються в мову QUEL і виконуються в INGRES.

HSDB [647] являє собою ССУБД, створену на основі реляційної системи Model 204 [648]. HSDB підтримує зведені таблиці й надає обмежений набір операцій над ними. Здатна виконувати процедури статистичного аналізу над реляційними і зведеними таблицями.

Розширена РМД [649]. Розширюється модель Кодда з метою представлення статистичних даних через запровадження «статистичної реляційної таблиці». Для неї розширюються реляційні операції і вводяться нові статистичні оператори. Пропонується мова запитів, що має подібні риси з QBE.

SYSTEM/K [650]. Об′єкт неорієнтована система управління базами знань створена на основі системи SQL/DS. Має широкі можливості з управління метаданими й обмежений перелік статистичних функцій.

GRAFSTAT [651]. Прикладна система призначена для аналізу даних із допомогою функцій прикладної статистики й графічного представлення результатів. Має інтерфейс із DB2 і SQL/DS через SQL.

SUBYL [652], PASTE [653], GPI [654], PEPIN-SICLA [655] є прикладами систем, що використовують традиційну СУБД, статистичний пакет і графічний пакет для створення системи управління статистичними даними.

2)Самостійно розроблені ССУБД. Вони мають шість підкатегорій відповідно до використовуваної моделі й мови запитів:

- Системи на базі реляційної моделі й реляційних мов запитів. Вони пропонують власні методи фізичної організації даних, засоби концептуального моделювання, придатні для ССУБД, а також можливості використання агрегуючих функцій у мовах запитів. До них належать:

RAPID [656] і CAS SBD [657] – використовують реляційну алгебру;

ABE [658] – використовує реляційне обчислення;

SIR/SQL [659], GENISYS [660], CANTOR [661] - використовують SQL.

У [662] представлено мову запитів статистичної обробки неповної інформації.

У системі July [663] використовується універсальний реляційний інтерфейс для інтерпретації статистичних запитів.

У статті [664] описана статистична модель даних та її застосування в СБД.

- Системи на базі ієрархічної і мережевої моделей. Прикладами є: SIR/DBMS [665], TPL і TPLDCS [666], BROWSE [667].

- Формальні розширення реляційної моделі: ABE [658], SSDL [668], SSDB [669].

- ССУБД і мови з графічним зовнішнім інтерфейсом. Системи даної категорії мають графічні двомірні або табличні мови запитів. Прикладами є: SUBJECT [621], GRASS [623], ABE [658], GUIDE [670], STBE [671], ALDS [672], GRASP [673].

- Природномовний інтерфейс користувача: LIDS 86 [674].

- Мови запитів, що обчислюють агреговану інформацію з темпоральних даних. Приклади: TQUEL [675], HQUEL [676], TBE [677], TEER [678], розширена реляційна алгебра Тансела [679].

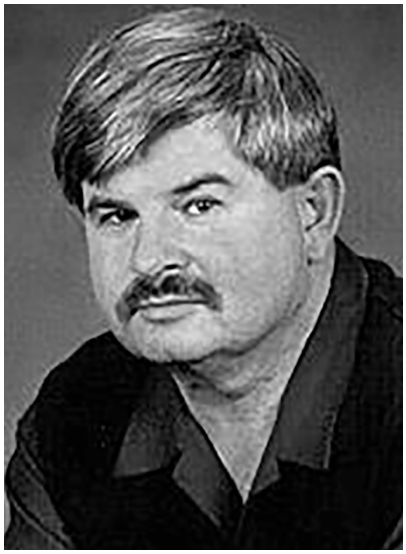
Підсумовуючи, відзначимо, що на підставі аналізу літератури можна стверджувати, що в розвиток дисципліни «статистичні бази даних» (принаймні на початковому етапі) значний внесок зробили турецько-американські вчені Зехра Мерал Озсойоглу (Zehra Meral Ozsoyoglu) і Гюльтекін Озсойоглу (Gultekin Ozsoyoglu).

Мерал Озсойоглу

Мерал Озсойоглу спеціалізується на базах даних. 2011 року вона отримала звання «Дійсний член ACM» (ACM Fellow) за «великий внесок у системи управління базами даних». А 2018 року отримала премію ACM SIGMOD Contributions Award за «віддане служіння співтовариству баз даних». У нагороді згадується її діяльність як головного редактора ACM Transactions on Database Systems і Proceedings of the VLDB Endowment, а також як голови програмного комітету конференції VLDB та Симпозіуму з принципів систем баз даних.

Сховища даних. Завдання збору інформації з різних джерел не є новим. Наприкінці минулого сторіччя поширилася концепція створення сховищ даних (Data Warehouse).

Термін «сховище даних» (DWH – Data Warehouse) уперше з'явився на початку 1980-х років, коли дослідники IBM Пол Мерфі (Paul Murphy) й Баррі Девлін (Barry Devlin) розробили сховище бізнес-даних. Проте основоположником сховищ даних вважається Вільям Х. Інмон (William H. Inmon), який почав дослідження в цій царині 1983 року, а власне концепція сховищ даних була викладена ним 1990 року в монографії [682]. Ця монографія стала своєрідною біблією сховищ даних і започаткувала розвиток їх індустрії. Інмон організував першу конференцію, вперше створив тематичний розділ у журналі.



Вільям Інмон

Ще одним фундатором справедливо вважається Ральф Кімбелл (Ralph Kimball). Він є одним із перших архітекторів сховищ даних, і його методологія (відома також як просторове моделювання або методологія Кімбелла) стала фактичним стандартом у системі підтримки прийняття рішень. 1990 року компанія Red Brick System, заснована Кімбеллом, розробила Red Brick Warehouse – компактну стандартну реляційну базу даних на основі SQL для додатків DWH і бізнес-аналітики. Його монографія [683] є бестселером і нині.

Ральф Кімбелл

Попри те, що ці вчені часом притримувалися протилежних поглядів на DWH, вони заснували і суттєво збагатили науку DWH. Порівняння поглядів на DWH цих двох учених наведено в [684] й багатьох інших працях дослідників їхнього творчого доробку.

Дослідниками – новаторами у DWH, які опублікували свої монографії в середині 90-х років, є також Брекетт [685], Гілл і Рао [686], По [687].

За Інмоном сховище даних – це предметно орієнтована, інтегрована, хронологічно послідовна і незмінна (постійна) колекція даних, створена для підтримки процесу прийняття рішень керівництвом [682]. Також вважається, що в широкому сенсі сховище даних – це сукупність технологій, котрі дозволяють керівництву приймати рішення швидше і якісніше, а тому вони є складовими автоматизованих систем підтримки прийняття рішень.

DWH передбачають інтеграцію гетерогенних (неоднорідних) БД. Однак тут є відмінність від традиційного підходу. Останній передбачає створення оболонок та посередників, що приводять стандартні запити до вигляду, який сприймається кожною з інтегрованих БД. Натомість у DWH інформація з численних гетерогенних джерел попередньо перетворюється, інтегрується і зберігається в єдиному сховищі даних.

Основним завданням традиційних БД, що отримали назву операційних, є ефективне виконання транзакцій із урахуванням активного оновлення БД, аби підтримувати її цілісність. Ці БД належать до класу систем оперативної обробки транзакцій (online transaction processing OLTP). З іншого боку, системи DWH не передбачають динамічного оновлення. Для них немає проблеми підтримання цілісності й призначені вони для користувачів, які здійснюють аналіз даних для прийняття рішень. Системи такого класу отримали назву систем оперативної аналітичної обробки (online analytical processing – OLAP).

Термін OLAP було введено Едгаром Коддом у його публікації в журналі Computerworld 1993 року [688]. В цій статті він визначив OLAP як засіб динамічного аналізу, синтезу й консолідації великих обсягів багатовимірних даних, сформулював концептуальні положення OLAP, описав архітектуру, виділив фундаментальні компоненти й за аналогією до 12 правил для реляційних баз даних запропонував 12 принципів аналітичної обробки.

На початку 1995 року Найджел Пендс (Nigel Pendse), не згідний із критеріями Кодда, запропонував альтернативні 5 правил приналежності систем до категорії OLAP [689]. Вони дістали назву «тести FASMI» - абревіатура від перших букв слів фрази “Fast Analysis of Shared Multidimensional Information” (швидкий аналіз спільно використовуваної багатовимірної інформації). Це визначення є також доволі популярним серед спеціалістів OLAP.

Найджел Пендс

Зрештою, рада OLAP (OLAP Council), створена 1995 року, дала таке розгорнуте визначення OLAP:

«Оперативна аналітична обробка (OLAP) – це категорія програмних технологій, які дозволяють аналітикам, менеджерам і керівникам мати уявлення про дані за рахунок швидкого, погодженого, інтерактивного доступу до представленої в різному вигляді інформації, перетвореної з вихідних даних, щоб вони таким чином усвідомлювали реальний стан речей на підприємстві» [690].

Архітектура DWH. Пропонувалося чимало різних архітектурних рішень DWH [691-702], кожне з яких має свої специфічні особливості. В праці [703] здійснено аналіз 73 архітектур DWH. На базі пропозицій, висловлених у [691, 692] подано узагальнену архітектуру DWH. DWH має трирівневу архітектуру.

Нижчий рівень являє собою БД DWH. Вона підтримує обрану модель даних DWH і надає засоби ведення цієї БД.

Середній рівень виконує функції OLAP. Зазвичай він представлений наступними чотирма типами [693, 694]:

Реляційний OLAP (ROLAP) – розширена реляційна СУБД, яка відображає операції багатовимірної моделі даних у стандартні операції реляційної алгебри.

Багатовимірний OLAP (MOLAP) – СУБД, яка безпосередньо підтримує багатовимірну модель даних та її операції.

Гибрідний OLAP (HOLAP) поєднує в собі якості попередніх двох видів.

Спеціалізований SQL – сервер має розвинуті можливості мови запитів SQL

для роботи з DWH-схемами (зірка, сніжинка, сузір′я фактів) у режимі лише читання.

Зовнішній рівень містить інструментальні засоби підтримки прикладних завдань DWH, включно з:

- бізнес аналітикою (business intelligence),
- оперативною аналітичною обробкою (OLAP),
- інтелектуальним аналізом даних (data mining),
- системами підтримки прийняття рішень (dicision support systems),
- мовами запитів і створення звітів.

Окрім цих трьох рівнів архітектури DWH включає:

- Репозиторій метаданих, який містить інформацію про дані DWH;
- Вітрини даних (data marts), що містять підмножину корпоративних даних, цікавих для певної групи користувачів.
- Засоби управління і контролю.
- Інструментальні засоби завантаження даних із зовнішніх джерел (бази даних, файли, електронні таблиці тощо) у БД DWH.

Ця компонента дістала назву ETL (Extract, Transform, Load). Вона виконує функції одержання даних із джерел, їх перевірки і очистки, перетворення до належного вигляду, інтеграції й завантаження або оновлення БД DWH [704]. Концепція ETL виникла в 1970-х роках у зв′язку з використанням централізованих репозиторіїв даних. Але лише в кінці 1980-х і початку 1990-х років вона набула великої популярності в зв′язку з появою DWH.

Моделі DWH. Із архітектурної точки зору виділяють наступні три типи моделей DWH [705]:

-корпоративне сховище із консолідованими даними, взятими з кількох операційних джерел – це DWH усієї корпорації [706];

-вітрина даних – містить підмножини корпоративних даних;

-віртуальне сховище – це множинність поглядів (views) операційних БД [707, 708].

Також існує точка зору [709], що архітектура DWH включає: архітектуру моделі даних, процесну архітектуру, інформаційну архітектуру, технологічну та ресурсну архітектури.

Вітрина даних (data mart). Концепція вітрин даних була запропонована Forrester Research ще 1991 року. Це предметно спрямована база даних, яка зазвичай містить дані одного з напрямків діяльності компанії. Вона орієнтована на користувачів однієї робочої групи або департаменту. У вітрині інформація зберігається оптимізовано з точки зору вирішення конкретних задач.

Існує три типи вітрин даних, які відрізняються залежно від їх відношення до сховища даних.

Залежні вітрини даних – це сегменти в корпоративному сховищі даних. Цей низхідний підхід починається зі збереження всіх бізнес даних в одному центральному місці. Новостворені вітрини даних отримують певну підмножину первинних даних щоразу за потреби аналізу.

Незалежні вітрини даних діють як автономна система, яка не покладається на сховище даних. Аналітики можуть діставати дані щодо конкретного предмету або бізнес-процесу із внутрішніх або зовнішніх джерел даних, обробляти їх, а відтак зберігати в репозиторії вітрини даних доти, доки вони знадобляться групі.

Гібридні вітрини даних об′єднують дані з існуючих сховищ та з інших операційних джерел. Цей уніфікований підхід використовує швидкість і зручний інтерфейс низхідного підходу, а також пропонує інтеграцію незалежного методу на рівні підприємства.

Ідея об′єднати дві концепції – сховищ даних і вітрин даних, очевидно, належить Марку Демаресту (Marc Demarest) [710], який 1994 року запропонував об′єднати дві концепції та використати сховище даних як єдине інтегроване джерело даних для вітрин даних.

Для взаємодії між собою вітрини даних можуть об′єднуватися в мережу, тим самим створюючи віртуальне сховище даних.

Багатовимірна модель даних DWH. Куб даних. Було запропоновано безліч багатовимірних моделей даних. Їх класифікація, аналіз і порівняння наведені в праці [711]. Коротко опишемо одну з них, яка використовується найбільше, а саме, куб даних [712].

Куб даних передбачає моделювання і подання даних, використовуючи поняття багатовимірного простору. Куб даних визначається через поняття «факт» і «вимір».

Згідно [713] терміни «факт» і «вимір» виникли в кінці 1960-х років у результаті виконання спільного дослідницького проєкту корпорації General Mills і Дартмутського університету. В 1970-х роках маркетингові компанії AC Nielsen і IRI постійно вживали ці терміни для опису своїх агрегованих даних і прагнули використати просторові моделі для презентації аналітичної інформації.

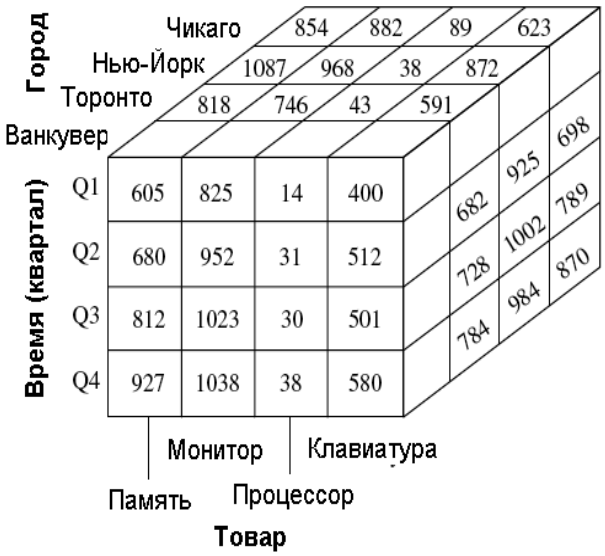
Вимір (dimension) – це характеристика, відносно якої представлено дані, що агрегуються. Використовуючи n вимірів, отримуємо n – мірний куб. Вимір – це вісь куба.

Вимір може ділитися на підвиміри. Приміром, вимір «країна» - на під виміри «області», а області – на «міста» тощо, таким чином утворюючи ієрархічну структуру виміру.

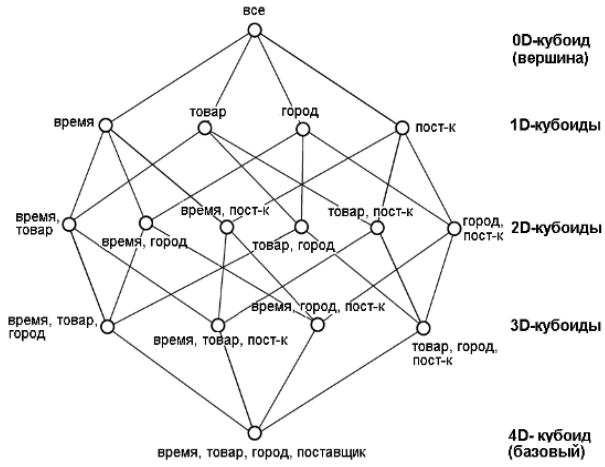
Факт – це характеристика, відносно якої представлені агреговані дані. Факт може мати властивості (атрибути).

Міра (measures) – це власне агреговані значення. Міри знаходяться в комірках кубу.

Багатовимірна модель має графічне зображення, що отримало назву кубоїда (cuboid) [714]. Кубоїд із найнижчим рівнем агрегованих даних (по всіх вимірах), називається базовим кубоїдом. На рисунку нижче подано трирівневий базовий кубоїд.



Із цього трирівневого кубоїда можна отримати три двомірні кубоїди шляхом агрегування даних по кожному з трьох вимірів. Із них – три одномірних і, зрештою, один нуль мірний кубоїд (кубоїд – вершина). Тобто одне агреговане значення всіх мір вихідного кубоїда. Така структура є решіткою кубоїдів (cuboids lattice) [715] і називається кубом. На рисунку нижче наводиться куб (решітка кубоїдів) для чотиривимірного базового кубоїда з вимірами: товар, час, місто, постачальник. Іноді кубоїди називають також кубами чи підкубами.



Операції над OLAP – кубами. Існує п′ять основних операцій над кубами OLAP:

Згортання (roll-up), що також називається узагальненням (drill-up). Приводить до агрегування куба даних чи то переміщенням вгору по ієрархічній структурі виміру (перехід від окремого поняття до загальнішого), чи-то видаленням виміру шляхом агрегування всіх мір цього виміру.

Розгортання (roll-down), що також має назву деталізація (drill-down). Операція протилежна згортанню/узагальненню – перехід від узагальнених даних до детальніших через переміщення вниз по ієрархічній структурі виміру чи введення нових вимірів.

Зріз (slice) – добування з куба підмножини комірок, пов′язаних із якимось певним значенням одного з його вимірів. Тобто отримуємо куб, де один з вимірів містить одне значення. Жодна агрегація мір не відбувається.

Фрагментація (dice) – є узагальненням зрізу. Із куба добувається підкуб, що має лише ті значення кожного з вимірів, які вказані в операції. Жодна агрегація мір не відбувається.

Обертання (pivot) – дозволяє змінювати просторову орієнтацію осей вимірів куба, обираючи найзручніше для аналітика представлення. В OLAP-технологіях куб – це передусім засіб візуалізації багатовимірних даних. Тому, використовуючи його, необхідно вирішувати задачу відображення інформації у зручному й інтерпретованому для людини вигляді.

Крім того, були запропоновані наступні додаткові операції.

Об′єднання (drill across) – дозволяє об′єднувати багато кубів, які мають один або більше спільних вимірів.

Проникнення (drill through) – дозволяє переходити від даних на нижньому рівні куба (базовий куб) до вихідних даних, звідки куб було добуто. Операція зазвичай використовується для визначення причини «викидів» у кубі даних.

Агрегуючі функції. Невід′ємна частина OLAP-моделі – завдання функцій агрегування. Оскільки мета OLAP – створення багаторівневої моделі аналізу, дані на всіх рівнях включно з базовим мають бути відповідно агреговані. По кожному виміру можливо задавати власну (і не одну) функцію агрегації.

Такі функції включають: функції агрегування, статистичні функції, функції ранжування Top N, Bottom N тощо. В [716] наведено класифікацію агрегуючих функцій з точки зору складності розпаралелювання.

У зв′язку із складністю структури куба опубліковано багато статей щодо його ефективної реалізації. Огляд досліджень у цій царині наведено в працях [715, 717]. Крім того, вичерпний огляд із реалізації ROLAP-кубів наведено в [718].

На закінчення відзначимо, що куб даних використовується не лише для представлення багатовимірних даних, а й інших складних типів даних. Серед них просторові, темпоральні, текстові, мультимедійні, мережеві і графічні [719, 720].

Багатовимірні бази даних (ББД) – це різновид БД, що створюється для сховищ даних і оперативної аналітичної обробки даних (OLAP). OLAP, що працюють з ББД, називаються багатомірними OLAP (MOLAP). Як правило, ББД використовують модель багатовимірних кубів для представлення висхідних даних. У [721] стверджується, що математичний апарат багатовимірних БД було розроблено видатним американським математиком Доном Нельсоном (Don Nelson) у 60-х роках на замовлення міністерства оборони США.

Концептуальні схеми DWH. За аналогією із ER-схемою концептуальної моделі ПЗ, прийнятою в традиційній технології проєктування реляційних OLTP-баз даних, у технології проєктування DWH було запропоновано наступні OLAP-схеми: зірка, сніжинка й сузір′я фактів [714].

Схема зірки. Найпопулярніша схема, яка містить:

-одну велику центральну таблицю фактів із даними по всіх мірах;

-безліч невеликих за розміром таблиць вимірів, по одній на кожен вимір. Ця таблиця містить відомості (атрибути) виміру. Графічне зображення цієї схеми нагадує зірку, де таблиці вимірів знаходяться радіально навколо таблиці фактів.

Схема сніжинки. Є узагальненням схеми зірки. В даному випадку, якщо таблиця вимірів містить багато «різнопланових» атрибутів (приміром, вона містить атрибути не лише країни, а й міст), така таблиця нормалізується, тобто розбивається на кілька «додаткових» таблиць (таблиць підвимірів). Граф результуючої схеми нагадує сніжинку.

Сузір′я фактів. Припускає існування багатьох таблиць фактів, що мають спільні таблиці вимірів. Графічно ця схема представлена безліччю зв′язаних схем зірок.

Враховуючи складність процесу концептуального моделювання DWH, було здійснено чимало досліджень із питання оцінки якості цього процесу, огляд яких подається в [42].

Методології проектування. Пропонуються такі три методології проєктування.

Проєктування знизу – вгору. Ця методологія висунута Кімбеллом і передбачає попереднє проєктування вітрин даних із конкретним тематичним напрямком. Остан-

ні представлені самостійними продуктами із наступним їх обслуговуванням в DWH.

Проєктування зверху – вниз. Запропонована Інмоном методологія передбачає спершу створення централізованого репозиторію DWH із використанням «нормалізованої» моделі даних ПЗ. Далі на основі DWH створюються вітрини даних для конкретних додатків або підрозділів підприємства.

Гібридне проєктування. Передбачає поєднання попередніх підходів і забезпечує всебічне й надійне проєктування.

Інструментальні засоби. Було зроблено багато інструментальних засобів DWH. За адресою https://www.guru99.com/top-20-etl-database-warehousing-tools.html наводиться стислий опис 26 найпопулярніших інструментальних систем класу DWH.

Активні DWH. На початку цього століття була висунута концепція активних DWH [723, 724] для того, щоб DWH підтримували автоматичнее прийняття рішень. В активних DWH розширюється технологія, яка є основою активних БД. А саме, вводяться «правила аналізу», що імітують роботу аналітика під час прийняття рішення. Водночас з'явилися перші комерційні продукти DWH із обмеженими можливостями активних правил [725, 726].

DWH реального часу [727]. Ця концепція розрахована на те, що вихідні дані надходять до DWH одразу, як тільки вони були породжені їхнім джерелом і стають доступними для аналізу. Про такі системи говорять, що вони є DWH «з нульовою затримкою». Популярність даної концепції сприяла тому, що багато виробників, включно з IBM [728] та Oracle [729], почали виробництво DWH цього класу. Стислий аналіз досліджень із цього напрямку наведено в [730].

Еволюція DWH. DWH уможливлюють збереження й аналіз даних за значний проміжок часу. Через те, що реальний світ, відображений у DWH, змінюється, те саме має відбуватися в DWH. Кімбалл, ймовірно, був першим, хто звернув на це увагу 1996 року й запропонував низку рішень [731]. З легкої руки Кімбалла ця проблема дістала назви «виміри, що повільно змінюються» (Slowly Changing Dimentions – SCD). Відтоді в цьому напрямку було здійснено чимало досліджень, стислий огляд деяких з них наведено в [732].

Темпоральні DWH. Темпоральні DWH містять такі ж структурні компоненти, що й традиційні DWH. А саме, виміри, ієрархії змін, факти й міри. Основною ж відмінністю є те, що в нетемпоральних DWH час може асоціюватися лише з фактами, які зазвичай представляють теперішній час (в термінах темпоральних БД). А в темпоральних DWH є можливість відслідковувати еволюцію вимірів, фактів і мір. Окрім того, темпоральні DWH, як і темпоральні БД, можуть бути бітемпоральними. Дослідження з темпоральних DWH охоплюють різні аспекти. Зокрема, темпоральні типи [733], концептуальне моделювання й проєктування [734], логічне моделювання й запити [735, 736], затримка в отриманні вимірів [737], багатовимірна агрегація [738], коректна агрегація за наявності змін у даних і структурі [739], еволюція багатовимірних схем [740]. У праці [741] подається огляд темпоральних DWH.

Просторові DWH. Просторові DWH (Spatial DWH – SDWH) виникли в зв'язку з бурхливим розвитком додатків, що мають відношення до оперування просторовими даними і передовсім географічних інформаційних систем (geographic information system – GIS). SDWH – це такі DWH, які уможливлюють оперування просторовими об'єктами задля підтримки просторово-орієнтованої ділової активності й прийняття рішень.

У праці [742] вперше було введено поняття просторового OLAP (SOLAP), яке відображає застосування методів інтелектуального аналізу щодо обробки просторових даних. У працях [743, 744] було введено поняття просторових вимірів і запропоновано їх класифікацію. В статті [745] пропонується розширення концептуальної багатовимірної моделі просторовими вимірами, ієрархіями і мірами, а також введенням у модель топологічних зв'язків і операторів. Були досліджені способи представлення просторових мір для геометричних об'єктів із використанням системи координат [742, 743, 745, 746] і сукупності точок [744].

SOLAP застосовують переважно до дискретних просторових даних, однак чимало складних задач GID-аналізу передбачають використання безперервних просторових даних, що зазвичай називаються просторовими полями. Просторові поля, або просто поля, описують фізичні явища, які безперервно змінюються в просторі або в часі. Як-от температура і тиск повітря, узвишшя землі, поширення буревію. Поля зазвичай представляються у вигляді функцій, які надають певні значення кожній точці простору.В зв'язку з цим здійснюються дослідження і розробки із створення польових DWH. Серед перших у цій царині праць була стаття [747], де пропонується куб даних із безперервними вимірами. В статті [748] також пропонується багатовимірна модель даних із безперервними вимірами і з набором операцій, котра може застосовуватися для OLAP-аналізу польових даних. У працях [749-751] подана модель і алгебра для роботи з просторово-часовими безперервними полями та їх використання для OLAP-аналізу просторових даних.

Було здійснено чимало інших досліджень із SDWH. Добрим вступом до просторових DWH є стаття [752]. У статті [753] подано аналітичний огляд фундаментальних методів і концепцій, що становлять основу просторових DWH.

SQL і OLAP. 1995 року група дослідників на чолі з Джеймсом Греєм запропонувала розширення мови SQL – фразу CUBE BY, завдання якої - створення OLAP-кубів [754]. CUBE BY створює групування за всіма можливими комбінаціями вказаних у ньому вимірів, із різними рівнями агрегації даних. Ця ідея була сприйнята в SQL:1999.

В SQL:1999 з'явилися можливості роботи з OLAP-кубами. Для цього фраза GROUP BY була розширена фразами ROLLUP, CUBE і GROUPING SETS, а також додана функція GROUPING.

Фраза ROLLUP створює умови для багаторівневого ієрархічного групування за вказаними в ній стовпчиками й створює проміжні суми (subtotals) у відповідності зі збільшуваним рівнем агрегації. Від найдеталізованіших рівнів представлення даних до більш узагальнених сум.

Фраза CUBE дозволяє в одній команді вирахувати всі можливі комбінації проміжних сум. Висловлюючись термінами решітки кубів, вказані в цій фразі стовпчики формують базову таблицю, й для неї створюється решітка. Фраза CUBE здатна генерувати інформацію, необхідну для перехресних звітів (cross – tabulation reports), в одному запиті.

Фраза GROUPING SETS формує результати угрупувань за вказаними в ній стовпчиками й об'єднує їх в одну таблицю. Інакше кажучи, вона еквівалентна конструкції UNION ALL до вказаних груп.

Функція GROUPING повертає істину в разі, якщо вказаний вислів є статистичним (тобто має підсумкове значення), і – неправду, якщо вислів нестатистичний.

References

439. Rothnie J.B. Jr, Bernstein P.A., Fox S., Goodman N., Hammer M., Landers T.A., Reeve C.L., Shipman D.W., Wong E. Introduction to a system for distributed databases (SDD-1). ACM Trans. on Database Syst. 1980;5(1):1–17.
440. Bernstein P.A., Shipman D.W., Rothnie J.B. Concurrency Control in a System for Distributed Databases (SDD-1) ACM Transactions on Database Systems, Vol. 5, No. 1, March 1980, Pages 19-51.
441. Hammar M., Shipman D. Reliability mechanism for SDD-1: a system for distributed database. ACM Trans. Database Syst., 5 (4) (Dec. 1980), pp. 431-466
442. Bernstein P.A., Goodman N., Wong E., Reeve C.L., Rothnie J.B. Query Processing in a System for Distributed Databases (SDD-1). ACM Transactions on Database Systems, Vol. 6, No. 4, December 1961, Pages 602-625
443. Selinger P.G. An architectural overview of R*: a distributed database management system. In: Proceedings of the 5th Berkeley Workshop on Distributed Data Management and Computer Networks; 1981, p. 187.
444. Williams R., Daniels D., Haas L., Lapis G., Lindsay B., Ng P. Obermarck R., Selinger P., Walker A., Wilms P., Yost R. R*: An Overview of the Architecture.IBM Research Report RJ3325, IBM Research Laboratory, San Jose, CA, Dec. 1981.

445. Lohman G.M., Mohan C., Haas L.M., Daniels D., Lindsay B.G., Selinger P.G., Wilms P.F. Query Processing in R*. Lohman G.M. et al. (1985) Query Processing in R*. In: Kim W., Reiner D.S., Batory D.S. (eds) Query Processing in Database Systems. Topics in Information Systems. Springer, Berlin, Heidelberg, pp. 31-47
446. Daniels D. et al. An Introduction to Distributed Query Compilation in R*. Distributed Data Bases (ed. H.-J. Schneider): Proc. 2nd Int. Symposium on Distributed Data Bases. - New York, N.Y.: North-Holland, 1982.
447. Stonebraker M.R., Neuhold E.J. A Distributed Data Base Version of INGRES // Proc. 2nd Berkley Conf. On Distributed Data Management and Computer Networks. — Lowrence Berkley Laboratory, May 1977.
448. Epstein R., Stonebraker M., Wong E. Distributed Query Processing in a Relational Database System // Proc. 1978 ACM SIGMOD Int. Conf. on Management of Data. — Austin, Tex. — May-June 1978.
449. Stonebraker M. The design and implementation of distributed INGRES. The INGRES Papers, Reading; 1986, p. 187–196.
450. Adiba M.E., Andrade J.M., Fernandez F., Gia Toan Nguyen. POLIPHEME: An experience in distributed database system design and implementation. Proc. of Int. Simposium on Distributed Data Bases. Paris, France, 1980, pp. 475-479
451. Epstein R., Stonebraker M., Wong E. Distributed Query Processing in a Relation Data Base System. SIGMOD '78: Proceedings of the 1978 ACM SIGMOD international conference on management of dataMay 1978 Pages 169–180.
452. Davenport R.A. Distributed database technology — a survey. Computer Networks (1976), Volume 2, Issue 3, 1978, Pages 155-167,
453. Rothnie J.B., Goodman N. A Survey of Research and Development in Distributed Database Management. VLDB '77: Proceedings of the third international conference on Very large data bases - Volume 3October 1977 Pages 48–62
454. Sheth A.P., Larson J.A. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput Surv. 1990;22(3):183–236.
455. Stonebraker M., Aoki P.M., Pfeffer A, Sah A, Sidell J, Staelin C, Yu A. Mariposa: a wide-area distributed database system. VLDB J. 1996;5(1):48–63.
456. Date C.J. What is a Distributed Database System? In: Date C. J. Relational Database Writings 1985-1989. — Reading, Mass.: Addison-Wesley, 1990.
457. Heimbigner D., McLeod D. "A Federated Architecture for information management". ACM Transactions on Information Systems, 1985,.Volume 3, Issue 3. pp. 253–278.
458. Sheth A.P., Larson J.A. "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases". ACM Computing Surveys, 1990, Vol. 22, No.3. pp. 183–236.
459. Masood N., Eaglestone B. "Component and Federation Concept Models in a Federated Database System". Malaysian Journal of Computer Science, 2003, 16 (2): 47–57.
460. Sheth A., Larson J.A. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. ACM Computing Surveys 1990, 22(3):183--236
461. Litwin W., Mark L., Roussopoulos N. Interoperability of multiple autonomous databases. ACM Comput. Surv. 1990; 22(3):267–293.
462. Wiederhold G. Mediators in the architecture of future information systems. IEEE Comput. 1992;25(3): 38–49.
463. Risch T., Josifovski V., Katchaounov T. (2004) Functional Data Integration in a Distributed Mediator System. In: Gray P.M.D., Kerschberg L., King P.J.H., Poulovassilis A. (eds) The Functional Approach to Data Management. Springer, Berlin, Heidelberg. pp. 211-238
464. Gribble S.D., Halevy A.Y., Ives Z.G., Rodrig M., Suciu D. What Can Database Do for Peer-to-Peer? In Processing of Int'l Workshop on the WEB and Databases (WebDB), 2001, pp. 31-36
465. Bonifati A., Chrysanthis P.K., Ouksel A.M., Sattler K.-U. Distributed databases and peer-to-peer databases: Past and present, ACM SIGMOD Record, 2008, 37(1): 5-11
466. Beng Chin Ooi, Kian-Lee Tan, guest editors. Introduction: special section on peer-to-peer-based data management. IEEE Trans Knowl Data Eng. 2004;16(7):785–786.

467. Sacca D., Wiederhold G. Database partitioning in a cluster of processors. ACM Trans Database Syst. 1985;10(1):29–56.
468. Yoshida M., Mizumachi K., Wakino A., Oyake I., Matsushita Y. Time and cost evaluation schemes of multiple copies of data in distributed database systems. IEEE Trans. Softw. Eng. 1985; 11(9) :954–958.
469. Ceri S., Negri M., Pelagatti G. Horizontal data partitioning in database design. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 1982. p. 128–136.
470. Ceri S., Pelagatti G. Distributed databases: principles and systems. New York: McGraw-Hill; 1984.
471. Navathe S., Ceri S., Wiederhold G., Dou J. Vertical partitioning of algorithms for database design. ACM Trans Database Syst. 1984;9(4):680–710.
472. McCormick W.T., Schweitzer P.J., White T.W. Problem decomposition and data reorganization by a clustering techniqiue. Oper Res. 1972;20(5): 993–1009
473. Shikha Mehta, Parul Agarwal, Prakhar Shrivastava, Jharna Barlawala, Differential bond energy algorithm for optimal vertical fragmentation of distributed databases, Journal of King Saud University - Computer and Information Sciences, 2018,
474. Chu W.W. Optimal file allocation in a multiple computer network. IEEE Trans Comput. 1969;- 18(10):885–889.
475. Apers P.M. Data allocation in distributed database systems. ACM Trans Database Syst. 1988;13(2): 263–304.
476. Bell D.A. Difficult data placement problems. Comput J. 1984;27(4):315–320.
477. Chang C.C, Shieh J.C. On the complexity of file allocation.problem. In: Proceedings of the International.Conference on the Foundations of Data Organization; 1985. p. 177–181.
478. Brunstrom A., Leutenegger S.T, Simha R. Experimental evaluation of dynamic data allocation strategies in a distributed database with changing workloads. In: Proceedings of the 40) 4th International Conference on Information and Knowledge Management; 1995. p.395–402.
479. Karlapalem K., Ng M.P. Query-driven data allocation algorithms for distributed database systems. In: Proceedings of the 8th International Conference Database and Expert Systems Applications; 1997. p. 347–356.
480. Bernstein P.A., Hadzilacos V., Goodman N. Concurrency control and recovery in database systems. Reading: Addison Wesley; 1987.
481. Gray J., Helland P., O'Neil P., Shasha D. The dangers of replication and a solution. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 1996. p. 173–182.
482. Breitbart Y., Komondoor R., Rastogi R., Seshadri S., Silberschatz A. Update propagation protocols for replicated databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 1999. p. 97–108.
483. Saito Y., Shapiro M. Optimistic replication. ACM Comput Surv. 2005;37(1):42–81.
484. Lin Y., Kemme B., Patiño-Martínez M., Jiménez-Peris R. Middleware based data replication providing snapshot isolation. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 2005. p. 419–430.
485. Wiesmann M., Schiper A. Comparison of database replication techniques based on total order broadcast. IEEE Trans Knowl Data Eng. 2005;17(4):551–566
486. Corbett J.C., Dean J., Epstein M., Fikes A., Frost C., Furman J.J., Ghemawat S., Gubarev A., Heiser C., Hochschild P., Hsieh W.C., Kanthak S., Kogan E., Li H., Lloyd A., Melnik S., Mwaura D., Nagle D., Quinlan S., Rao R., Rolig L., Saito Y., Szymaniak M., Taylor C., Wang R., Woodford D. Spanner: Google's globally distributed database. ACM Trans Comput Syst. 2013;31(3):8:1-8:22.
487. Mahmoud H.A., Nawab F., Pucher A., Agrawal D., El Abbadi A. Low-latency multi-datacenter databases using replicated commit. Proc VLDB Endow. 2013;6(9):661–672.
488. Satyanarayanan M., Kistler J.J., Kumar P., Okasaki M.E., Siegel E.H., Steere D.C. Coda: a highly available file system for a distributed workstation environment. IEEE Trans Comput. 1990;39(4):447–459.
489. Terry D.B., Theimer M., Petersen K., Demers A.J., Spreitzer M., Hauser C. Managing update conflicts in Bayou, a weakly connect-

ed replicated storage system. In: Proceedings of the 15th ACM Symposium on Operating System Principles; 1995. p. 172–183.
490. Sivasubramanian S., Szymaniak M., Pierre G., van Steen M. Replication for web hosting systems. ACM Comput Surv. 2004;36(3):291–334.
491. Lv Q., Cao P., Cohen E., Li K., Shenker S. Search and replication in unstructured peer-to-peer networks. In: Proceedings of the 16th Annual International Conference on Supercomputing; 2002. p. 84–95.
492. Budhiraja N, Marzullo K, Schneider FB, Toueg S. The primary-backup approach. In: Mullender S, editor. Distributed systems. 2nd ed. Reading: Addison Wesley; 1993. p. 199–216.
493. Schneider F.B. Replication management using the state-machine approach. In: Mullender S, editor. Distributed systems. 2nd ed. Reading: Addison Wesley; 1993. p. 169–198.
494. Almeida S., Leitão J., Rodrigues L.E.T. Chainreaction: a causal+ consistent datastore based on chain replication. In: Proceedings of the 8th ACM SIGOPS/EuroSys European Conference on Computer Systems; 2013. p. 85–98.
495. Sovran Y., Power R., Aguilera M.K., Li J. Transactional storage for geo-replicated systems. In: Proceedings of the 23rd ACM Symposium on Operating System Principles; 2011. p. 385–400.
496. Gray J. Notes on data base operating systems. In: Advanced Course: Operating Systems; 1978. p. 393–481.
497. Ho G.S, Ramamoorthy C.V. Protocols for deadlock detection in distributed database systems. IEEE Trans Softw Eng. 1982;8(6):554–557.
498. Stonebraker M. The design and implementation of distributed ingres. In: The INGRES papers: anatomy of a relational database system; 1986. p. 187–96.
499. Menascé D.A., Muntz R. Locking and deadlock detection in distributed data bases. IEEE Trans Softw Eng. 1997;5(3):195–202.
500. Mohan C., Lindsay., Bruce G., Obermarck R. Transaction management in the R* distributed database management system. ACM Trans Database Syst. 1986;11(4):378–396.
501. Abonamah A.A,. Elmagarmid A. A survey of deadlock detection algorithms in distributed database systems. In: Advances in distributed and parallel processing. System paradigms and methods, vol. 1; 1994. p. 310–341.
502. Elmagarmid A.K. A survery of distributed deadlock algorithms. ACM SIGMOD Rec. 1986;15(3):37–45.
503. Knapp E. Deadlock detection in distributed databases. ACM Comput Surv. 1987;19(4): 303–328.
504. Singhal M. Deadlock detection in distributed systems. Computer. 1989;22(11):37–48.
505. Krivokapic N, Kemper A, Gudes E. Deadlock detection in distributed database systems: a new algorithm and a comparative performance analysis. VLDB J. 1999;8(2):79–100.
506. Roesler M., Burkhard W.A., Cooper K.B. Efficient deadlock resolution for lock-based concurrency control schemes. In: Proceedings of the 18th International Conference on Distributed Computing Systems; 1998. p. 224–233.
507. Bracha G., Sam T. Distributed deadlock detection. Distrib Comput. 1985;2(3):127–138.
508. Chandy K.M., Lamport L. Distributed snapshots: determining global states of distributed systems. ACM Trans Comput Syst. 1986;3(1):63–75.
509. Bernstein P.A., Goodman N., Wong E., Reeve C.L., Rothnie Jr.J.B. Query processing in a system for distributed databases (SDD-1). ACM Trans Database Syst. 1981;6(4):602–625.
510. Epstein RS, Stonebraker M, Wong E. Distributed query processing in a relational data base system. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 1978. p. 169–180.
511. Stonebraker M. The design and implementation of distributed INGRES. In: Stonebraker M, editor. The INGRES papers. Reading: Addison-Wesley; 1986. 4) Williams R., Daniels D., Hass L., Lapis G., Lindsay B., Ng P., Obermarck R., Selinger P., Walker A., Wilms P., Yost R. R*: an overview of the architecture. IBM Research Lab, San Jose, Technical Report RJ3325; 1981.
512. Williams R., Daniels D., Hass L., Lapis G., Lindsay B., Ng P., Obermarck R., Selinger P., Walker A., Wilms P., Yost R. R*: an overview of the architecture. IBM Research Lab, San Jose, Technical Report RJ3325; 1981.
513. Haas L.M., Selinger P.G., Bertino E., Daniels D., Lindsay B.G., Lohman G.M., Masu-

naga Y., Mohan C., Ng P., Wilms P.F., Yost R.A. R*: a research project on distributed relational DBMS. IEEE Database Eng Bull. 1982;5(4):28–32.
514. Wong E. Retrieving dispersed data from SDD-1: a system for distributed databases. In: Proceedings of the 2nd Berkeley Workshop on Distributed Data Management and Computer Networks; 1977. p. 217–235.
515. Yu C.T. and Chang C.C. Distributed query processing. ACM Comput. Surv., 16(4):399–433, 1984.
516. Kossmann D. The state of the art in distributed query processing. ACM Comput Surv. 2000;32(4):422–469.
517. Urhan T., Franklin M.J., Amsaleg L. Cost based query scrambling for initial delays. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 1998. p. 130–141.
518. Stonebraker M., Devine R., Kornacker M., Litwin W., Pfeffer A., Sah A., Staelin C. An economic paradigm for query processing and data migration in Mariposa. In: Proceedings of the 3rd International Conference Parallel and Distributed Information Systems; 1994. p. 58–67.
519. Ceri S., Pelagatti G. Distributed databases principles and systems. New York: McGraw-Hill; 1984.
520. Eswaran K.P., Gray J.N., Lorie R.A., Traiger I.L. The notion of consistency and predicate locks in a database system. Commun ACM. 1976;19(11):624–633.
521. Gray J.N. The transaction concept: virtues and limitations. In: Proceedings of the 7th International Conference on Very Data Bases; 1981. p 144–154.
522. Spector A.Z., Schwarz P.M. Transactions: a construct for reliable distributed computing. ACM Operat Syst Rev. 1983;17(2):18–35
523. Stearns R.E., Rosenkrantz D.J. Distributed database concurrency controls using beforevalues. SIGMOD ‘81: Proceedings of the 1981 ACM SIGMOD international conference on Management of data, 1981, pp 74–83
524. Boral H., Gold I. Towards a self-adapting centralized concurrency control algorithm. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 1984. p. 18–32.
525. Lausen G. Concurrency control in database systems: a step towards the integration of optimistic methods and locking. In: Proceedings of the ACM Annual Conference; 1982. p. 64–68.
526. Salem K., Garcia-Molina H., Shands J. Altruistic locking. ACM Trans Database Syst. 1994;19(1):17–165.
527. Kung H.T. Robinson J.T. “On Optimistic Methods for Concurrency Control”. ACM Transactions on Database Systems, Vol. 6, No. 2, 1981, pp. 213-226.
528. Rahm E. Concepts for Optimistic Concurrency Control in Centralized and Distributed Database Systems. IT Informationstechnik, (in German), 1988, vol. 30, no. 1, pp. 28-47.
529. Thomasian A. Distributed optimistic concurrency control methods for high-performance transaction processing IEEE Transactions on Knowledge and Data Engineering, 1998, 10(1):173 - 189
530. Bernstein P., Goodman N. [1980] “Timestamp-Based Algorithms for Concurrency Control in Distributed Database Systems,” in VLDB ‘80: Proceedings of the sixth international conference on Very Large Data Bases - Volume 6October, 1980, pp. 285–300
531. Bernstein PA., Goodman N., RothnieJ.B.Jr., Papadimitriou C.H. Analysis of serializability of SDD1: a system of distributed databases (the fully redundant case). IEEE Trans. On Software Engineering, SE-4: 3 (1978), pp. 154-168.
532. Reed D.P. “Implementing Atomic Actions on Decentralized Data,” TOCS, 1:1, February 1983, pp. 3–23.
533. Reed D.P. Naming and synchronization in a decentralized computer system. Ph. D. Thesis, MIT, Cambridge, Mass., 1977
534. Thomasian A. Concurrency Control: Methods, Performance, and Analysis, ACM Computing Surveys, 1998, 30(1):70-119
535. Ozkarahan E. Database Machines and Database Management. Englewood Cliffs, N.J.; Prentice-Hall, 1986 - 636 p.
536. DeWitt D.J., Hawthorn P.B. A performance evaluation of data base machine architectures. In: Proceedings of the 7th International Conference on Very Data Bases; 1981. p. 199–214.
537. Boral H., DeWitt D.J., Wilkinson W.K. Performance evaluation of four associative disk

designs Information Systems, Volume 7, Issue 1, 1982, Pages 53-64
538. Boral H., DeWitt D. Database machines: An idea whose time has passed? A critique of the future of database machines. In Proceedings of the 1983 Workshop on Database Machines. H.-O. Leilich and M. Missikoff, Eds., Springer-Verlag, 1983, pp. 16--187
539. Slotnik D.L. "Logic per Track Devices" in Advances in Computers, Vol. 10., Frantz Alt, Ed., Academic Press, New York, 1970, pp. 291 - 296. TODS, Vol 1, No. 3. September 1976.
540. Parker J.L. "A Logic per Track Retrieval System," IFIP Congress, 1971. J.L. Parker, "A Logic per Track Retrieval System", Proc. IFIP Congress 1971, pp. TA-4-146 to TA-4-150
541. Minsky N., "Rotating Storage Devices as Partially Associative Memories" Proc. 1972 FJCC. N. Minsky: Rotating Storage Devices as Partially Associative Memories, FJCC 1972, AFIPS Conf. Proc., pp. 587–595
542. Parhami B. "A Highly Parallel Computing System for Information Retrieval" Proceedings of the Fall Joint Computer Conference, 1972. pp. 681-690
543. Ozkarahan E.A., Schuster S.A., Smith K.S. RAP: An Associative Processor for Data Base Management. Proc. AFIPS 44, NCC, 1975, pp. 379-387.
544. Su S.Y.W., Lipovski G.J."CASSM: A Cellular System for Very Large Data Bases", VLDB '75: Proceedings of the 1st International Conference on Very Large Data BasesSeptember 1975 Pages 456–472
545. Lin S.C., Smith D.C.P., Smith J.M. "The Design of a Rotating Associative Memory for Relational Database Applications," TODS Vol. 1, No. 1, pages 53 - 75, Mar. 1976.
546. Kannan K. "The Design of a Mass Memory for a Database Computer," Proc. Fifth Annual Symposium on Computer Arcliecture. Palo Alto, CA. April 1978, pp. 44-51
547. Leilich H.-O., Stiege G., Zeidler H.Ch. "A Search Processor for Data Base Management Systems" VLDB '78: Proceedings of the fourth international conference on Very Large Data Bases - Volume 4, September 1978, Pages 280–287
548. Schuster S.A., Nguyen, H.B., Ozkarahan, E.A. K.C. Smith, "RAP.2 - An Associative Processor for Databases and its Applications," IEEE Transactions on Computers, C-28, No. 6, June 1979. pp. 446-458
549. DeWitt D.J., "DIRECT - A Multiprocessor Organization for Supporting Relational Database Management Systems," IEEE Transactions on Coiputers. June 1979, pp. 395-406.
550. Madnick S.E. "The Infoplex Database Computer: Concepts and Directions," Proceedings of the IEEE Computer Conference, Feb. 1979l, pp. 168-176
551. Hell W. "RDBM - A Relational Database Machine: Archtecture and Hardware Design," Proceedings of the 6th Workshop on Computer Architecture for Non-Numeric Processing, June 1981,
552. Missikoff M. "An Overview of the project DBMAC for a relational machine," Proceedings of the 6th Workshop on Computer Architecture for Non-Numeric Processing, Hyeres, France, June 1981.
553. DeWitt D.J., Gray J. Parallel database systems: the future of high performance database systems. Commun ACM. 1992;36(6):85–98.
554. Hurson A.R., Miller L.L., Pakzad S.H., Eich M.H., Shirazi B. Parallel architectures for database systems. Advances in Computers, Vol. 28, 1989, pp. 107-151.
555. Stonebraker M. "The Case for Shared Nothing," Database Engineering, Vol. 9, No. 1, 1986. pp. 4-9
556. Stonebraker M., Katz, R.H., Patterson, D.A., Ousterhout, J.K., The Design of XPRS, Fourteenth Int. Conf. on Very Large Data Bases, (Los Angeles, 1988), Morgan Kaufmann, 1988, pp. 318–330.
557. Bergsten B., Couprie, M., Lopez, M., DBS3: A Parallel Data Base System for Shared Store (Synopsis), in Issues, Architectures, and Algorithms (Proc. of the 2nd Int. Conf. on Parallel and Distributed Information Systems (PDIS 1993), San Diego, 1993), IEEE Comput. Soc., 1993, pp. 260–262.
558. Graefe G., Volcano—An Extensible and Parallel Query Evaluation System, IEEE Trans. Knowledge Data Engineering, 1994, vol. 6, no. 1, pp. 120–135.
559. Strickland J.P., Uhrowczik, P.P., Watts, V.L., IMS/VS: An Evolving System, IBM Systems J., 1982, vol. 21, no. 3, pp. 490–510.
560. Linder B., Oracle Parallel RDBMS on Massively Parallel Systems, in Issues, Architec-

tures, and Algorithms (Proc. of the 2nd Int. Conf. on Parallel and Distributed Information Systems (PDIS 1993), San Diego, 1993), IEEE Comput. Soc., 1993, pp. 67–68.
561. Dubova N., Supercomputers nCube, Otkrytye sistemy, 1995, no. 2, pp. 42–47.
562. Kronenberg N.P., Levy, H.M., Strecker, W.D., VAXclusters: A Closely-Coupled Distributed System, ACM Trans. Comput. Systems, 1986, vol. 4, no. 2, pp. 130–146.
563. Nick J.M., Moore B.B., Chung J.-Y., Bowen N.S., S/390 Cluster Technology: Parallel Sysplex, IBM Systems J., 1997, vol. 36, no. 2, pp. 172–201.
564. Teradata: DBC/1012 Data Base Computer Concepts & Facilities, Teradata Corp. Document No. C02-0001-00, 1983.
565. Dewitt D.J., Ghandeharizadeh S., Schneider D.A., Bricker A. Hsiao H.-I., Rasmussen R. "The Gamma Database Machine Project," IEEE Knowledge and Data Engineering, Vol. 2, No. 1, March, 1990, pp. 44-62
566. Tandem Performance Group, "A Benchmark of Non-Stop SQL on the Debit Credit Transaction," Proceedings of the 1988 SIGMOD Conference, Chicago, IL, June 1988.
567. Alexander W., Copeland G.P. Process And Dataflow Control In Distributed Data-Intensive Systems. Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, June 1-3, 1988. ACM Press, 1988. P. 90-98
568. Lorie R., Daudenarde J., Hallmark G., Stamos J., Young H., "Adding Intra-Transaction Parallelism to an Existing DBMS: Early Experience", IEEE Data Engineering Newsletter, Vol. 12, No. 1, March 1989., pp. 2–8.
569. Gibbs J, " Massively Parallel Systems, Rethinking Computing for Business and Science," Oracle, 1991, Vol. 6, No.1
570. Engler, S., Glasstone R., Hasan W., Parallelism and Its Price: A Case Study of NonStop SQL/MP, ACM SIGMOD Record, 1995, vol. 24, no. 4, pp. 61–71.
571. Clay D. Informix Parallel Data Query (PDQ), in Issues, Architectures, and Algorithms (Proc. of the 2nd Int. Conf. on Parallel and Distributed Information Systems (PDIS 1993), San Diego, 1993), IEEE Comput. Soc., 1993, pp. 71–72.
572. Page J., A Study of a Parallel Database Machine and Its Performance: The NCR/Teradata DBC/1012. Advanced Database Systems, Lecture Notes in Computer Science (Proc. of the 10th British Natl. Conf. on Databases. BNCOD 10, Aberdeen, 1992), Springer, 1992, vol. 618, pp. 115–137.
573. Baru C.K. et al. DB2 Parallel Edition, IBM System J., 1995, vol. 34, no. 2, pp. 292–322.
574. Sokolinsky L.B. Survey of Architectures of Parallel Database Systems (Rus). Programming and Computer Software volume 30, No 6, pages 337–346 (2004)
575. Copeland G.P., Keller T., A Comparison of High-Availability Media Recovery Techniques, Proc. of the 1989 ACM SIGMOD Int. Conf. on Management of Data (Portland, 1989), ACM, 1989, pp. 98–109.
576. Graefe G., Query Evaluation Techniques for Large Databases, ACM Computing Surv., 1993, vol. 25, no. 2, pp. 73–169.
577. Hua K.A., Lee C., Peir J.-K., Interconnecting Shared-Everything Systems for Efficient Parallel Query Processing, Proc. First Int. Conf. on Parallel and Distributed Information Systems (PDIS 1991) (Miami Beach, 1991), IEEE-CS, 1991, pp. 262–270.
578. Pramanik S., Tout W.R. The NUMA with Clusters of Processors for Parallel Join, IEEE Trans. Knowledge Data Eng., 1997, vol. 9, no. 4, pp. 653–666.
579. Bouganim L., Florescu D., Valduriez P. Dynamic Load Balancing in Hierarchical Parallel Database Systems, Proc. 22th Int. Conf. on Very Large Data Bases (VLDB'96) (Mumbai, India, 1996), Morgan Kaufmann, 1996, pp. 436–447.
580. Xu Y., Dandamudi S.P. Performance Evaluation of a Two-Level Hierarchical Parallel Database System, Proc. Int. Conf. Computers and Their Applications, Tempe, Arizona, 1997, pp. 242–247.
581. Korneev V.V. Parallel'nye vychislitel'nye sistemy (Parallel Computing Systems), Moscow: Nolidzh, 1999.
582. Shmidt V. IBM SP2 Systems (Rus), Otkrytye Sistemy, 1995, no. 6, pp. 53–60.
583. Shnitman V. Fault-Tolerant Servers ServerNet (Rus), Otkrytye Sistemy, 1996, no. 3, pp. 5–11.
584. Sokolinsky L.B. Organization of Parallel Query Processing in Multiprocessor Database Machines with Hierarchical Architecture, Programmirovanie, 2001, no. 6, pp. 13–29.

585. Velicanu M., Litan D., Mocanu (Virgolici) A.-M., 2010. "Some Considerations about Modern Database Machines," Informatica Economica, Academy of Economic Studies - Bucharest, Romania, vol. 14(2), pages 37-44.
586. Eric G. Oracle and storage IOs, explanations and experience at CERN," 17th International Conference on Computing in High Energy and Nuclear Physics, Prague, Czech Republic, March 2009, pp. 21 – 27.
587. Chock M., Cardenas A., Klinger A. Database structure and manipulation capabilities of a picture database management system (PICDMS). IEEE ToPAMI, 6(4):484–492, 1984
588. Maier D., Vance B. A call to order. In PODS '93: Proceedings of the twelfth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, 1993, pp. 1–16
589. Baumann P. Management of Multidimensional Discrete Data. VLDB Journal, Special Issue on Spatial Database Systems, 1994, Vol 4, No. 3, pp. 401–444
590. Baumann P. A Database Array Algebra for Spatio-Temporal Data and Beyond, 4th International Workshop on Next Generation Information Technologies and Systems (NGITS '99), July 5–7, 1999, Zikhron Yaakov, Israel, Lecture Notes on Computer Science 1649, Springer Verlag, pp. 76 – 93.
591. Baumann P., Dehme A., Furtado P., Ritsch R., Widmann N. The Multidimensional Database System RasDaMan. Conference: SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA. ACM SIGMOD Record, 1998, Vol. 27, No. 2, pp 575–577
592. EarthServer: The EarthServer Initiative. www.earthserver.eu
593. Baumann P., Holsten S. (2011) A Comparative Analysis of Array Models for Databases. In: Kim T. et al. (eds) Database Theory and Application, Bio-Science and Bio-Technology. BSBT 2011, DTA 2011. Communications in Computer and Information Science, vol 258. pp 80-89 Springer, Berlin, Heidelberg.
594. Baumann, P., Misev, D., Merticariu, V., Bang Pham Huu. Array databases: concepts, standards, implementations. Journal of Big Data, vol 8, No. 1 (2021).
595. Tomlin D. A Map Algebra. Harvard Graduate School of Design, 1990.
596. Mennis, J., Viger, R., Tomlin, C.D.: Cubic Map Algebra Functions for Spatio-Temporal Analysis. Cartography and Geographic Information Science, Vol. 32, No. 1, 2005, pp. 17-32.
597. Ritter G, Wilson J, Davidson J. Image Algebra: An Overview. Computer Vision, Graphics, and Image Processing. Vol. 49, No. 3, 1990, pp. 297-331.
598. Marathe A, Salem K. A language for manipulating arrays. In: Proceedings of the 23th International Conference on Very Large Data Bases; 1997. p. 46–55.
599. Libkin, L., Machlin, R., Wong, L.: A query language for multidimensional arrays: design, implementation and optimization techniques. Proc. ACM SIGMOD'96, Montreal, Canada/ ACM SIGMOD Record, 1996, vol. 25, No. 2, pp. 228–239
600. Machlin R. Index-based multidimensional array queries: safety and equivalence. In L. Libkin, editor, PODS, pp. 175–184. ACM, 2007.
601. van Ballegooij A.R., de Vries A.P., Kersten M. RAM: Array processing over a relational DBMS. Technical Report INS-R0301, CWI (March 2003)
602. van Ballegooij A.R. RAM: A multidimensional array DBMS. In W. Lindner, M. Mesiti, C. Turker, Y. Tzitzikas, and A. Vakali, editors, EDBT Workshops, volume 3268 of Lecture Notes in Computer Science, pp. 154–165. Springer, 2004.
603. Cornacchia R., Heman S., Zukowski M., de Vries A., Boncz P. Flexible and efficient IR using array databases, VLDB Journal. 7(1): 151–168.
604. Stonebraker M., Brown P., Poliakov A., Raman S. (2011) The Architecture of SciDB. In: Bayard Cushing J., French J., Bowers S. (eds). Proceedings of the 23rd International Conference on Scientific and Statistical Database Management; 2011 pp 1-16
605. Kersten M.L, Zhang Y., Ivanova M., Nes N. SciQL, a query language for science applications. Proceedings, EDBT/ICDT 2011 Workshop on Array Databases: Uppsala, Sweden, March 25, 2011, pp
606. Cheng, Y., Rusu, F. Formal representation of the SS-DB benchmark and experimental evaluation in EXTASCID. Distrib Parallel Databases, 2015, vol. 33, No. 3, pp. 277–317

607. Cheng Y., Rusu, F. Astronomical data processing in EXTASCID. SSDBM: Proceedings of the 25th International Conference on Scientific and Statistical Database Management, 2013 Article No.: 47, pp. 1–4
608. Tollefsen, Andreas Forø (2013) PostGIS 2.0 og Raster, Kart og Plan 73(3), pp. 159–164.
609. Teradata. Multidimensional array options. - https://docs.teradata.com/r/VrFCOAaniAIfrJsA51oQJA/ZMY8sE8cSytuSPtp8QnuFA
610. Oracle: GeoRaster. - http://docs.oracle.com/cd/B19306_01/appdev.102/b14254/geor_intro.htm.
611. Becla J., Lim K.T. Report from the first Workshop on Extremely Large Databases, Data Science Journal, 2008, Vol. 7, pp. 1-13
612. Stonebraker M., Brown P., Poliakov A., Raman S. (2011) The Architecture of SciDB. In: Bayard Cushing J., French J., Bowers S. (eds). Proceedings of the 23rd International Conference on Scientific and Statistical Database Management; 2011 pp. 1-16
613. Bauman National Library. SciDB. - https://ru.bmstu.wiki/SciDB
614. Ivanova M, Kersten M.L, Manegold S. Data vaults: a symbiosis between Database technology and scientific file repositories. Proc. Intl. Conference on Scientific and Statistical Database Management (SSDBM). Athens. 2012, pp. 485-:494.
615. Zhang Y, Kersten M.L, Ivanova M, Nes N. SciQL, bridging the gap between science and relational DBMS. In: Desai B.C, Cruz I.F, Bernardino J, editors. Proceedings of the 15th Symposium on International Database Engineering and Applications; 2011. pp. 124–133.
616. Baumann P., Stamerjohanns H. (2014) Towards a Systematic Benchmark for Array Database Systems. In: Rabl T., Poess M., Baru C., Jacobsen H.A. (eds) Specifying Big Data Benchmarks. pp 94-102.
617. "ISO/IEC DIS 9075-15 Information technology -- Database languages -- SQL -- Part 15: Multi-dimensional arrays (SQL/MDA)"
618. Furtado P., Baumann P. Storage of multidimensional arrays based on arbitrary tiling. In Proceedings of the 15th International Conference on Data Engineering, pp. 328–336. IEEE Computer Society, 23-26 March 1999
619. Srivastava J., Ngo H.Q. Statistical Databases. Technical Report TR 99-009, 1999, Department of Computer Science and Engineering, University of Minnesota. - https://conservancy.umn.edu/bitstream/handle/11299/215365/99-009.pdf?sequence=1&isAllowed=y
620. Michalewicz Z. (ed.) Statisticai and Scientific Databases. Market Cross House, Cooper Street, Chichester, West Sussex, PO19 1EB, 11991, 544 p.
621. Chan P., Shoshani A. SUBJECT: A directory driven system for large statistical databases. In VLDB '81: Proceedings of the seventh international conference on Very Large Data Bases - Volume 7, 1981, pp. 553–563
622. Su S. SAM: A semantic association model for corporate and scientific-statistical databases. Journal of Information Science, pp. 151–199, 1983.
623. Rafanelli M., Ricci F.L. A visual interface for browsing and manipulating statistical entities. In Proceedings of the Fifth International Conference on Scientific and Statistical Database Management, pp. 1990, 163–182,
624. Battista G.D., Batini C. Design of statistical databases: a methodology for the conceptual step. The Journal of Information Systems, vol. 13, no. 4, pp. 407–422, 1988
625. Rafanelli M., Shoshani A. STORM: A statistical object representation model. In Proceedings of the Fifth International Conference on Scientific and Statistical Database Management, pp. 14–29, 1990.
626. Rafanelli M., F.L. Ricci, "Mefisto: A functional model for statistical entities," IEEE Transactions on Knowledge and Data Engineering, vol. 5, No. 4, pp. 670–681, Aug. 1993.
627. Ghosh S.P. (1989) Statistical relational model. In: Rafanelli M., Klensin J.C., Svensson P. (eds) Statistical and Scientific Database Management. SSDBM 1988. Lecture Notes in Computer Science, vol 339. Springer, Berlin, Heidelberg. pp. 338-355
628. Shoshani A., Kawagoe K. Temporal data management. In Proceedings of the Twenty-Second International Conference on Very Large Data Bases (VLDB), 1986 pp. 79–90.
629. Meo-Evoli L., Ricci F.L., Shoshani A., On the Semantic Completeness of Macro-Data Operators for Statistical Aggregation, SSDBM 1992, pp. 239-258.
630. Ozsoyoglu G., Ozsoyoglu Z.M., Matos V., Extending relational algebra and relational

calculus with set-valued attributes and aggregate functions," ACM Transactions on Database Systems, 1987, vol. 12, pp. 566–592.
631. Ozsoyoglu G., Ozsoyoglu Z.M., Malta F. A Language and a Physical Organization Technique for Summary Tables. SIGMOD, 1985: pp. 3-16.
632. Gray J., Bosworth A., Layman A., Pirahesh H. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. Data Mining and Knowledge Discovery, 1997, Vol. 1, No. 1, pp. 29–53
633. Agrawal R., Gupta A., Sarawagi S. Modeling Multidimensional Databases. ICDE '97: Proceedings of the Thirteenth International Conference on Data EngineeringApril, 1997, pp. 232–243.
634. Vassiliadis P. "Modeling multidimensional databases, cubes and cube operations," Proceedings. Tenth International Conference on Scientific and Statistical Database Management (Cat. No.98TB100243), 1998, pp. 53-62,
635. Gentle J.E., Bell J. Special Data Types and Operators for Statistical Data. IEEE Database Eng. Bull., 1984, Vol. 7, No. 1, pp. 34-37
636. Fortunato E., Rafanelli M., Ricci F., Sebastio A.,An algebra for statistical data. In SSDBM'86: Proceedings of the 3rd international workshop on Statistical and scientific database management, 1986, pp. 122–134
637. Bezenchek A., Rafanelli M., Tininini L. A data structure for representing aggregate data. In Proceedings: Eighth International Conference on Scientific and Statistical Database Systems, Stockholm, Sweden (P. P. Svensson and J. C. J. C. French, eds.),, IEEE Computer Society Press, 1996, pp. 22–31
638. van den Berg G.M., E. de Feber. Definition and use of meta-data in statistical data processing. In Proceedings of the 6th International Conference on Statistical and Scientific Management, (Ascona, Switzerland), 1992, pp. 290–306
639. Kent J.P., Schuerhoff M. Some thoughts about a metadata management system, in Proceedings of the 9th International Conference on scientific and Statistical Databases, (Olympia, WA), pp. 155–164, IEEE Press, Aug. 1997.
640. Westlake A. "A simple structure for statistical meta-data," in Proceedings of the 9th International Conference on scientific and Statistical Databases, (Olympia, WA), pp. 186–195, IEEE Press, Aug. 1997.
641. Ghosh S. P. Statistical Metadata. In Kotz-Johnson Encyclopedia of Statistical Science, Vol.8, John Wiley & Sons Inc. Publ., 1988
642. Signore M., Scanu M., Brancato G. Statistical metadata: a unified approach to management and dissemination. Journal of Official Statistics, 2015, Vol. 31, No 2, pp. 325-347
643. Tansel A. Query languages for statistical databases. Statistics and Compuing. 1995. Vol. 5, No. 1, pp. 59-72
644. Ozsoyoglu G., Ozsoyoglu, Z. M. Statistical database query languages. IEEE Transactions on Software Engineering. 1985, vol 11, No. 10, pp. 1071-1080.
645. Johnson R. Modeling summary data. In Proceedings of the ACM SIGMOD Conference, (Ann Arbor, Michigan), pp. 93–97, 1981.
646. Shoshani A. CABLE: A Chain-Based Language for the Entity-Relationship Model. Proceedings of the 1st International Conference on the Entity-Relationship Approach to Systems Analysis and Design, 1980, pp. 465–466
647. Ikeda H., Kobayashi Y. Additional facilities of a conventional DBMS to support interactive statistical analysis. In Proceedings of the 1st LBL Workshop on Statistical Database Management, Lawrence Berkeley Lab, Berkeley, CA, Dec. 1981, pp. 25–36
648. Computer Corporation of America. File Manager's Technical Reference Manual, Model 204 Database Management System. Computer Corporation of America, Cambridge, MA, 1979
649. Ghosh S.P. Statistical relational tables for statistical database management, IEEE Transactions of Software Engineering, 1986, vol. SE-12, No. 12, pp. 1106–1116.
650. Maier D., Cirilli C. SYSTEM/K: A knowledge based management system. In Proceedings of the Second Int. Workshop on Statistical Database Management, Los Altos, CA, Sept. 1983, pp. 287–294
651. Stein D.M. A database interface to an integrated dataanalysis and plotting tool. In Proceedings of the 3rd Inter-national Workshop on Statistical and Scientific Database Management, Luxemburg, 1986, pp. 98–106
652. Heiler S., Bergman R.F. SIBYL: An economist'sworkbench. In SSDBM'83: Pro-

ceedings of the 2nd International Workshop on Statistical Database Management, Los Altos, CA., 1983, pp. 73–79

653. Weiss S.E., Weeks P.L. PASTE-a tool to put application systems together easily. In SSDBM'83: Proceedings of the 2nd International Workshop on Statistical Database Management LosAltos CA, 1983, pp. 119–123
654. Hollabaugh L.A., Reinwald L.T. GPI: a statistical package/database interface. SSDBM'81: In Proceedings of the 1st International Workshop on Statistical Database Management MenloPark CA, 1981, pp. 78–87
655. Boufares P., Elkabbaj Y., Joiner G., Ounally H. Laversion SM90 du SGBD relationnel PEPIN. Journes SM90, Versailles, France, 1985
656. Turner M. T., Hammond R. Cotton P. A DBMS for large statistical databases. In VLDB '79: Proceedings of the fifth international conference on Very Large Data Bases - Volume 5, 1979, pp. 319–327
657. Johji S., Sato H. Statistical database research project in Japan and the CAS SDB project. In SSDBM'83: Proceedings of the 2nd international workshop on Statistical Database Management, 1983 pp. 325–330.
658. Klug A. ABE – a query language for constructing aggregates-by-example. In SSDBM'81: Proceedings of the 1st LBL Workshop on Statistical Database Management, 1981, pp. 190–205
659. Anderson G., Snider T., Robinson B., Toporek J. An integrated research support system for inter-package communication and handling large volumne output from statistical database analysis operation. In SSDBM'83: Proceedings of the 2nd international workshop on Statistical Database Management, 1983, pp. 104–110
660. Dintelman S.M., Maness A.T. An implementation of a query language supporting path expressions. In Proceedings of the 1982 ACM SIGMOD International Conference on Management of Data, (Orlando, Florida), 1982., pp. 87–93
661. Karasolo I., Sevenson P. An overview of CANTOR – a new system for data analysis. In SSDBM'83: Proceedings of the 2nd international workshop on Proceedings of the Second International Workshop on Statistical Database ManagementSeptember 1983 pp. 315–324
662. Chan C., Michalewicz Z. A query language capable of handling incomplete information and statistics. In SSDBM'86: Proceedings of the 3rd international workshop on Statistical and scientific database management, 1986, pp. 107–115
663. D'attri A., Ricci F.L. Interpretation of statsitical queries to relational databases. In SSDBM'1988: Proceedings of the 4th international conference on Statistical and Scientific Database Management, 1988, pp. 246–258
664. Chen M., McNamee L., Melkanoff M. A model of summary data and its applications in statistical databases. In SSDBM'1988: Proceedings of the 4th international conference on Statistical and Scientific Database Management, 1988, pp 356-387
665. Anderson G., Snider T., Robinson B., Toporek J. An integrated research support system for inter-package communication and handling large volumne output from statistical database analysis operation. In SSDBM'83: Proceedings of the Second International Workshop on Statistical Database Management, 1983, pp. 104–110.
666. Weiss W., Weeks P., Byrd P. Must we navigate through databases. In SSDBM'81: Proceedings of the 1st LBL Workshop on Statistical Database Management, Lawrence Berkeley Lab, Berkeley, CA, Dec. 1981, pp. 111–122
667. Hendrix G.G., Sacerdoti E.D., Sagalowicz D., Slocum J. Developing a natural language interface to a complex system. ACM Transactions on Database Systems, 1978, Vol. 3, No. 2,, pp. 105–147
668. Brown W., Navathe S., Su S. Complex data types and a data manipulation language for scientific and statistical databases. In SSDBM'83: Proceedings of the 2nd international workshop on Statistical Database Managemen, 1983, pp.188–195
669. Ozsoyoglu G., Ozsoyoglu Z.M. Features of a system for statistical databases. In SSDBM'83: Proceedings of the 2nd international workshop on Statistical Database Management, 1983, pp. 9–18.
670. Wong H.K.T., Kuo I. GUIDE: Graphical user interface for database exploration. Iin Proceedings of the 8th Conference on Very

Large Databases, Morgan Kaufman pubs. (Los Altos CA), McLeod and Villasenor, Mexico City, 1982, pp. 22-32
671. Ozsoyoglu Z. M., Ozsoyoglu G. Summary-table-by-example: A database query language for manipulating summary data. In Proceedings of the International Conference on Data Engineering, (Los Angeles, CA), 1984, pp. 193–202.
672. Thomas J., Hall D. ALDS project: Motivation, statistical database management issues, perspectives, and directions, In SSDBM'83: Proceedings of the 2nd international workshop on Statistical Database Management-September, 1983, pp.82–88
673. Catarci T., Santucci G. GRASP: A graphical system for statistical databases. In Proceedings of the Fifth International Conference on Scientific and Statistical Database Management, (Charlotte, NC), 1990, pp. 148–162
674. Sato H. A data model, knowledge base and natural language processing for sharing a large statistical database. Iin Proceedings of the 4th International Working Conference SSDBM on Statistical and Scientific Database Management, 1988, pp. 207–225
675. Snodgrass R. T., The temporal query language TQuel. In Symposium on Principles of Database Systems, 1984, pp. 204–213.
676. Tansel A.U., Arkun M.E. HQUEL, A query language for historical relational databases. In SSDBM'86: Proceedings of the 3rd international workshop on Statistical and scientific database management, 1986, pp. 135–142
677. Tansel A., Arkun M.E., Ozsoyoglu G. Time-by-example query language for historical databases. IEEE Transactions on Software Engineering (SE), 1989, vol. 15, No. 4, pp.464-478
678. Elmasri R., Kouramajian V. A temporal query language based on conceptual entities and roles. ER '92: Proceedings of the 11th International Conference on the Entity-Relationship Approach: Entity-Relationship Approach, 1992, pp. 375–388
679. Tansel A.U. A statistical interface for historical relational databases. In Proceedings of the Third International Conference on Data EngineeringFebruary, 1987, pp 538–546
680. Reznichenko V.A. Workig with windows in SQL (Rus). Software Engineering, 2011, vol. 7, No 3, pp. 35-48
681. Chandra P., Gupta M.K. Comprehensive survey on data warehousing research. International Journal of Information Technology. 10, pp. 217–224 (2018). https://doi.org/10.1007/s41870-017-0067-y
682. Inmon, W.H.'Building the data warehouse', 5th Edition, John Wiley & Son. 2005
683. Kimball R., Ross M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, Third Edition. John Wiley & Sons, Inc. 2013. 600 p.
684. Breslin M. Data Warehousing Battle of the Giants: Comparing the Basics of the Kimball and Inmon Models. In Business Intelligence Journal. 2004. pp. 6-20
685. Brackett M.H.The Data Warehouse Challenge: Taming Data Chaos. John Wiley & Sons, 1996, 579 pages.
686. Gill S.H., Rao P.C. The Official Client/Server Computing Guide to Data Warehouse. QUE Corporation, 1996, 382 pages.
687. Poe V. Building a Data Warehouse for Decision Support. Prentice Hall. 1995
688. Codd E.F. Providing OLAP to User-Analysts: An IT Mandate // Computerworld. — T. 27, № 30
689. Pendse N. What is OLAP? - http://dssresources.com/papers/features/pendse04072002.htm
690. Ponniah P. 'Data warehousing fundamentals', John Wiley & Sons, 2001, 516 p.
691. Han J., Kamber M.,Pei J. Data Mining: Concepts and Techniques, 3rd ed.The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, 2011, 703 p.
692. Chaudhuri S., Dayal U. An Overview of Data Warehousing and OLAP Technology. ACM SIGMOD Record, Volume 26, Issue 1, March 1997 pp 65–74. - https://doi.org/10.1145/248603.248616
693. Jensen C.S, Pedersen T.B, Thomsen C. Multidimensional databases and data warehousing. Synthesis lectures on data management. San Rafael: Morgan Claypool; 2010. 111 p.
694. Vaisman A, Zimányi E. Data Warehouse Systems: Design and Implementation (Data-Centric Systems and Applications) 2014th Edition. Springer; 2014.
695. Muhammad Arif, Ghulam Mujtaba. A Survey: Data Warehouse Architecture. International Journal of Hybrid Information Technology Vol.8, No. 5 (2015), pp. 349-356.

696. Astriani W.,Trisminingsih R. Extraction, Transformation, and Loading (ETL) module for hotspot spatial data warehouse using Geokettle. Procedia, Environmental Science, Elsevier, The 2nd International Symposium on LAPAN-IPB Satellite for Food Security and Environmental Monitoring 2015, pp 626-634
697. Chaudhary S., Murala D.P., Srivastav V.K. (2011) 'A critical review of data warehouse', Global Journal of Business Management and Information Technology, Volume(1):No.(2), pp. 95-103.
698. Oliveira B., Belo O. (2015) A Domain-Specific Language for ETL Patterns Specification in Data Warehousing Systems. In: Pereira F., Machado P., Costa E., Cardoso A. (eds) Progress in Artificial Intelligence. EPIA 2015. Lecture Notes in Computer Science, vol 9273. Springer, Cham. pp 597-602 https://doi.org/10.1007/978-3-319-23485-4_60
699. Data Warehouse Architecture, Concepts and Components. - https://www.guru99.com/data-warehouse-architecture.html
700. Data Warehouse Architecture: Types, Components, & Concepts. - https://www.astera.com/type/blog/data-warehouse-architecture/
701. Enterprise Data Warehouse: Concepts and Architecture. - https://www.altexsoft.com/blog/enterprise-data-warehouse-concepts/
702. Bhadresh Pandya, Dr. Sanjay Shah. Proposed Local Data Mart Approach for Data Warehouse Architecture. International Journal of Emerging Technology and Advanced Engineering. 2014. Vol. 4, No. 2. pp. 101-104
703. Yang Q., Ge M. Helfert M. Analysis of Data Warehouse Architectures: Modeling and Classification. In Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019), pages 604-611.
704. Kimball R., Caserta J. The Data Warehouse ETL Toolkit. Wiley Publ., 2004, 526 p.
705. Chandra P., Gupta M.K. Comprehensive survey on data warehousing research. International Journal of Information Technology.
706. Scabora L.C., Brito J.J., Ciferri R.R., Ciferri C.D.D.A. Physical data warehouse design on NoSQL databases – OLAP Query Processing over HBase. Proc. 18th Intern. Conf. SCITEPRESS. 2016, pp. 111–118. DOI: 10.5220/0005815901110118.10, pp. 217–224
707. Khan F.A., Ahmad A., Imran M., Alharbi M., Jan B. Efficient data access and performance improvement model for virtual data warehouse. Sustainable cities and society. 2017, vol. 35, pp. 232–240. DOI: 10.1016/j.scs.2017.08.003.
708. Gupta A., Mumick I.S. 'Maintenance of materialized views: problems, techniques, and applications', IEEE Data Engineering Bulletin, Special Issue on Materialized Views and Data Warehousing, 1995, Vol.18, No. 2, pp. 3-18
709. Sachin Chaudhary, Devendra Prasad Murala, V.K. Srivastav. A Critical Review of Data Warehouse. Global Journal of Business Management and Information Technology. 2011, Vol. 1, No.2, pp. 95-103
710. Demarest, "Building The Data Mart", DBMS Magazine, 1994. — Vol 7, No.8. — p. 44—50.
711. Pedersen T.B, Jensen C.S. Multidimensional data modeling for complex data. In: Proceedings of the 15th International Conference on Data Engineering; 1999. p. 336–345.
712. Vassiliadis P. Modeling multidimensional databases, cubes and cube operations. In: Proceedings of the 10th International Conference on Scientific and Statistical Database Management; 1998. p. 53–62.
713. Kimball R., Ross M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, Third Edition. John Wiley & Sons, Inc. 2013, 600 p.
714. Han J., Kamber M.,Pei J. Data Mining: Concepts and Techniques, 3rd ed.The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, 2011, 703 p. - УЖЕ ЕСТЬ ВЫШЕ
715. Harinarayan V, Rajaraman A, Ullman J.D. Implementing data cubes efficiently. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 1996. p. 205–216.
716. Sanjay Goil and Alok Choudhary. High performance OLAP and data mining on parallel computers. Center of Parallel and Distributed Computing Technical Report TR9705, 1997
717. Morfonios K., Ioannidis Y. Cube Implementations. In Encyclopedia of Database Systems, Ling Liu, M. Tamer Özsu Editors. pp. 710-716.

718. Morfonios K, Konakas S, Ioannidis Y, Kotsis N. ROLAP implementations of the data cube. ACM Computing Surveys, 2007. vol. 39, No. 4, Article 12, 53 pages/
719. Pedersen TB. Managing complex multidimensional data. In: Aufaure M-A, Zimányi E, editors. Business intelligence – second European summer school, eBISS 2012. Brussels: Springer LNBIB; 2013, 15–21 July 2012, Tutorial Lectures.
720. Vaisman A, Zimányi E. Data warehouse systems – design and smplementation. Springer; 2014.
721. Multidimensional DBMS. - https://tadviser.com/index.php/Article:Multidimensional_DBMS
722. Gosain A., Heena. Literature Review of Data model Quality metrics of Data Warehouse. International Conference on Intelligent Computing, Communication & Convergence (ICCC-2015). Procedia Computer Science 48 (2015) 236 – 243
723. Schrefl M, Thalhammer T. On Making Data Warehouses Active. In M. Mohania and A. Min Tjoa, editors, DaWaK 2000: Proceedings of the Second International Conference on Data Warehousing and Knowledge Discover, Greenwhich, London (UK), September 4-6, 2000. Springer LNCS, pp. 34–46
724. Thalhammer T, Schrefl M, Mohania M. Active data warehouses: complementing OLAP with analysis rules. Data & Knowledge Engineering, 2001, Vol. 39, No. 3, pp. 241–269.
725. Brobst S. Active data warehousing: a new breed of decision support. In: Proceedings of the 13th International Workshop on Data and Expert System Applications; 2002. p. 769–772.
726. Borbst S, Rarey J. The five stages of an active data warehouse evolution. Teradata Mag. 2001;3(1):38– 44.
727. Syed Ijaz Ahmad Bukhari: Real Time Data Warehouse. CoRR abs/1310.5254 (2013)
728. IBM Data Warehousing. -https://www.ibm.com/analytics/us/en/data-management/data-warehouse.
729. Best practices for Real-time Data Warehousing. An oracle white paper. 2014. http://www.oracle.com/us/products/middleware/data-integration/realtime-datawarehousing-bp-2167237.pdf
730. Mohania M., Nambiar U., Tam H., Schrefl M., Vincent M. Active, Real-Time, and Intellective Data Warehousing. In Encyclopedia of Database Systems, Ling Liu, M. Tamer Özsu Editors. pp 41-49
731. Kimball R. Slowly changing dimensions. DBMS Mag. 1996;9(4):14.
732. Eder J., Koncilia C., Wiggisser K. Data Warehouse Maintenance, Evolution, and Versioning. In In Encyclopedia of Database Systems, Ling Liu, M. Tamer Özsu Editors. pp. 884-890
733. Bruckner R, Min Tjoa A. Capturing delays and valid times in data warehouses: towards timely consistent analyses. J. Intell. Inf. Syst. 2002;19(2):169–190.
734. Malinowski E, Zimányi E. Advanced data warehouse design: from conventional to spatial and temporal applications. Berlin/Heidelberg: Springer; 2008.
735. Ahmed W, Zimányi E, Wrembel R. Temporal data warehouses: logical models and querying. In: Proceedings of the Journées francophones sur les Entrepôts de Données et l'Analyse en ligne, EDA. Editions Hermann; 2015. p. 33–48.
736. Mendelzon A, Vaisman A. Time in multidimensional databases. In: Rafanelli M, editor. Multidimensional databases: problems and solutions. Hershey: Idea Group; 2003. p. 166–199.
737. Golfarelli M, Rizzi S. Managing late measurements in data warehouses. Int J Data Wareh Min. 2007;3(4):51–67.
738. Böhlen M, Gamper J, Jensen C. Towards general temporal aggregation. In: Proceedings of the 25th British National Conference on Databases; 2008. p. 257–169.
739. Golfarelli M, Lechtenbörger J, Rizzi S, Vossen G. Schema versioning in data warehouses: enabling cross-version querying via schema augmentation. Data Knowl Eng. 2006;59(2):435–459.
740. Ahmed W, Zimányi E, Wrembel R. A logical model for multiversion data warehouses. In: Proceedings of the 16th International Conference on Data Warehousing and Knowledge Discovery; 2014. p. 23–34.
741. Golfarelli M, Rizzi S. A survey on temporal data warehousing. Int J Data Wareh Min. 2009;5(1):1–17.
742. Rivest S, Bédard Y, Marchand P. Toward better support for spatial decision making: defining the characteristics of spatial on-line

analytical processing (SOLAP). Geomatica 2001;55(4):539–555.
743. Bédard Y, Merrett T, Han J. Fundaments of spatial data warehousing for geographic knowledge discovery. In: Miller H, Han J, editors, Geographic data mining and knowledge discovery. London: Taylor & Francis; 2001. p. 53–73.
744. Stefanovic N, Han J, Koperski K. Object-based selective materialization for efficient implementation of spatial data cubes. IEEE Trans Knowl Data Eng. 2000;12(6):938–958.
745. Malinowski E, Zimányi E. Representing spatiality in a conceptual multidimensional model. In: Proceedings of the 12th ACM Symposium on Advances in Geographic Information Systems; 2004. p. 12–22.
746. Bimonte S, Tchounikine A, Miquel M. Towards a spatial multidimensional model. In: Proceedings of the 8th ACM International Workshop on Data Warehousing and OLAP; 2005. p. 39–46.
747. Shanmugasundaram J, Fayyad U, Bradley P. Compressed data cubes for OLAP aggregate query approximation on continuous dimensions. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 1999. p. 223–232.
748. Ahmed TO, Miquel M. Multidimensional structures dedicated to continuous spatiotemporal phenomena. In: Proceedings of the 22nd British National Conference on Databases; 2005. p. 29–40.
749. Gómez L, Gómez S, Vaisman A. Analyzing continuous fields with OLAP cubes. In: Proceedings of the 14th ACM International Workshop on Data Warehousing and OLAP; 2011. p. 89–94.
750. Gómez L, Gómez S, Vaisman A. A generic data model and query language for spatiotemporal OLAP cube analysis. In: Proceedings of the 15th Interna tional Conference on Extending Database Technology; 2012. p. 300–11.
751. Gómez L, Gómez S, Vaisman A. Modeling and querying continuous fields with OLAP cubes. Int J Data Wareh Min. 2013;9(3):22–45.
752. A.A. Vaisman, Zimányi E. Spatial Datawarehousing. Encyclopedia of Database Systems, Ling Liu, M. Tamer Özsu Editors, Second Edition, 2018, pp. 3587-3592
753. Bédard, Y., T. Merrett & J. Han, 2001, Fundamentals of Spatial Data Warehousing for Geographic Knowledge Discovery, Geographic Data Mining and Knowledge Discovery, Taylor & Francis, Vol. Research Monographs in GIS, No. Chap. 3, p. 53-73
754. Gray J, Chaudhuri S, Bosworth A, Layman A, Venkatrao, M, Reichart D, Pellow F, Pirahesh H. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab and Sub-Totals. Data Mining and Knowledge Discovery, 1997, 1(1):29–54.

Отримано: 27.07.2021

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

УДК 004.042 https://doi.org/10.15407/pp2022.01.67

В.О.Тюрін, А.Ю. Дорошенко, О.В. Савчук

АНАЛІТИЧНЕ СХОВИЩЕ ДЛЯ ВЕЛИКИХ ПОТОКОВИХ ДАНИХ

Розроблено концепцію архітектури з організації аналітичного сховища даних на основі інфраструктури Google Cloud Platform (GCP). Проведено аналіз існуючих рішень у галузі безсерверних аналітичних сховищ. Проведено порівняльний аналіз із найбільш розповсюдженими існуючими рішеннями та здійснено експериментальне випробування розробленої концепції. Наведено рекомендації з організації сховища даних з можливістю підтримки подій із змінною схемою даних. Розроблено систему потокової передачі даних. Розроблену концепцію повністю реалізовано у GCP з метою проведення функціонального тестування.

Ключові слова: безсерверні, аналітичні сховища, BigQuery, AWS, GCP, ETL, повідомлення, потокова передача даних.

Вступ

У роботі описано концепцію архітектурного рішення щодо безсерверного (serverless) аналітичного сховища даних. Запропоновані шляхи реалізації концепції, що дають змогу створювати аналітичні сховища:

- швидкими (завдяки написанню мінімальної кількості коду);
- з доброю масштабованістю (левова частка відповідальності за масштабування передається хмарному провайдеру);
- легкими у підтримці;
- з високим рівнем захисту даних;
- з легкою інтеграцією до засобів прийняття рішень.

Наразі подібні системи мають надвисокий попит через те, що ціна сховищ стала відносно дешевою і компанії отримали змогу зберігати й використовувати для аналізу великі об'єми операційних даних [1-4]. Через те, що не усі бізнеси мають досвідчені відділи IT, багато з таких компаній зазнають невдач у процесі створення аналітичного сховища [5-6]. Створення концепції подібного сховища на базі одного з найкращих провайдерів хмарних послуг (GCP) може стати проривним як у технологічному плані, так і у плані ефективності бізнесових рішень.

Розмір фінансових інвестицій у галузь уже сягає мільярдів доларів [7-8], проте бракує саме науково-технологічних робіт та стандартизації й аналізу наявних рішень. Ця проблема існує тому, що галузь розвивається дуже швидко і ще не підхоплена університетами. Тому науково-технічна документація здебільшого створена самими розробниками систем або їх партнерами із впровадження, а кожна робота розглядає лише єдиний варіант власної реалізації. Основна мета таких робіт – це просування власного бізнесу, а не деталізований технічний аналіз запропонованих рішень.

Створений у даній роботі проєкт концепції несе науково-технічну новизну в порівнянні з існуючими рішеннями, що перевіряється на детальному аналізі й формулюванні незалежної оцінки й оформленні концепції. Ця робота також дає можливість комерційного використання, впровадження або консультації із впровадження концепції та супроводу його технічної реалізації.

У роботі розглянуті існуючі рішення та стандарти у сфері побудови хмарних аналітичних сховищ даних, здійснено їх порівняльний аналіз на основі практичних реалізацій та надана незалежна оцінка сформованій концепції.

Головна мета роботи – створення концепції аналітичних сховищ даних.

Для цього потрібно було розв'язати наступні задачі:

- розглянути існуючі рішення;
- провести аналіз;
- розглянути технічні можливості, які не були запропоновані;

© В.О.Тюрін, А.Ю. Дорошенко, О.В. Савчук, 2022

• оцінити економічну складову;

• запропонувати відповідне архітектурне рішення.

Задля реалізації системи, яка відповідала б заданій меті, були проведені теоретичні та практичні дослідження у п'ять етапів:

• базова теоретична підготовка проєкту, що включала в себе теоретичну підготовку по всім архітектурним та технологічним напрямкам, які належать до сфери побудови аналітичних сховищ даних;

• теоретична підготовка в розділах хмарних систем, поглиблення знань та практичних навичок у використанні різних провайдерів хмарних послуг, вивчення сервісів, котрі надаються у використання;

• технічна реалізація концепції;

• порівняльний аналіз;

• створення технічної документації.

Проєкт є безкоштовним для користувачів, проте на основі розробки може бути створена компанія, що буде надавати послуги із впровадження моделі. Цільовий сегмент компаній, які можуть використати цей науковий проєкт, дуже широкий. Фактично це може бути будь-яка компанія, яка виросла достатньо, щоб почати використовувати свої операційні дані для прийняття рішень.

Для технічної реалізації ідей концепції були обрані сучасні хмарні технології.

Огляд платформи AWS Serverless Data Lake Framework (SDLF)

AWS Serverless Data Lake Framework (SDLF) – це стандарт (готове архітектурне рішення), засноване на компонентах (ресурсах) екосистеми AWS.

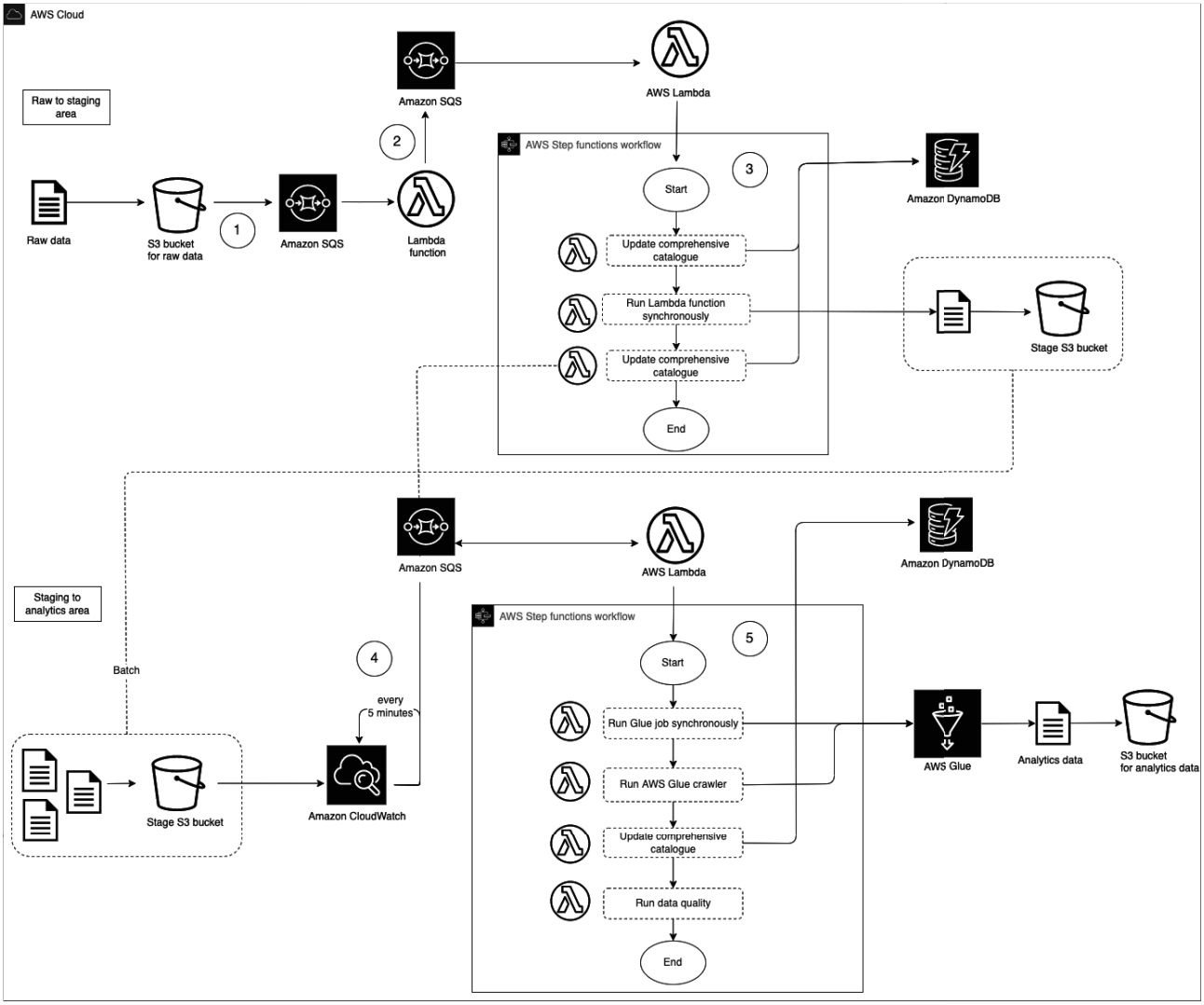


Рис. 1. Архітектурна схема AWS SDLF[10]

Це дозволяє швидко створювати «озеро даних» Data Lake, що відповідає сучасним потребам [9-10]. SDLF є головним конкурентом на ринку для розроблюваного стандарту.

Основними перевагами SDLF є:

• швидке розгортання (AWS Team надає шаблони готової інфраструктури);

• безсерверна архітектура (дозволяє значно скоротити необхідні навички і відповідно чисельність команди підтримки);

• гнучкість до змін (є можливість вставити свій код бізнес-логіки в поточну архітектуру).

До недоліків належать:

• повна залежність від AWS як провайдера інфраструктури (схема не може бути реалізована в інших хмарах);

• значні приховані затримки (latency) при високому навантаженні;

• незручний формат Landing Zone (файли із записами); у процесі реконсиляції чи під час пошуку проблеми знайти інформацію у Landing Zone майже неможливо;

• дуже складна модель контролю даних;

• невирішеність проблеми контролю схеми даних.

Використання Upsolver для IronSource DataLake

IronSource DataLake – це архітектурне рішення для побудови аналітичного сховища компанії. Компанія використала Upsolver – продукт, що полегшує створення подібних систем.

IronSource використовує Upsolver в найтяжчому для реализації місці в системі – контролі за схемами і взаємодії з продюсерами даних. Завдяки використанню ще одного продукту Kafka, керованого подіями, та регістру схем Schema Registry з'являється гарантія несуперечності схем постачальника та споживача даних.

Архітектура DataLake передбачає рівень RAW (LANDING) зберігання початкових даних – у даній архітектурі цей рівень, як і в SDLF реалізовано за допомогою Amazon S3 та файлів. Серед переваг цього підходу можна виділити дешевизну й зручність застосування одночасно декількох Query Engines до одних і тих самих даних. Проте водночас є й недоліки:

• дуже скаладна реконцеляція RAW Layer; тобто, за наявності певних проблем із несуперечністю даних перевірка даних на рівні RAW стає непростою задачею для інженерів;

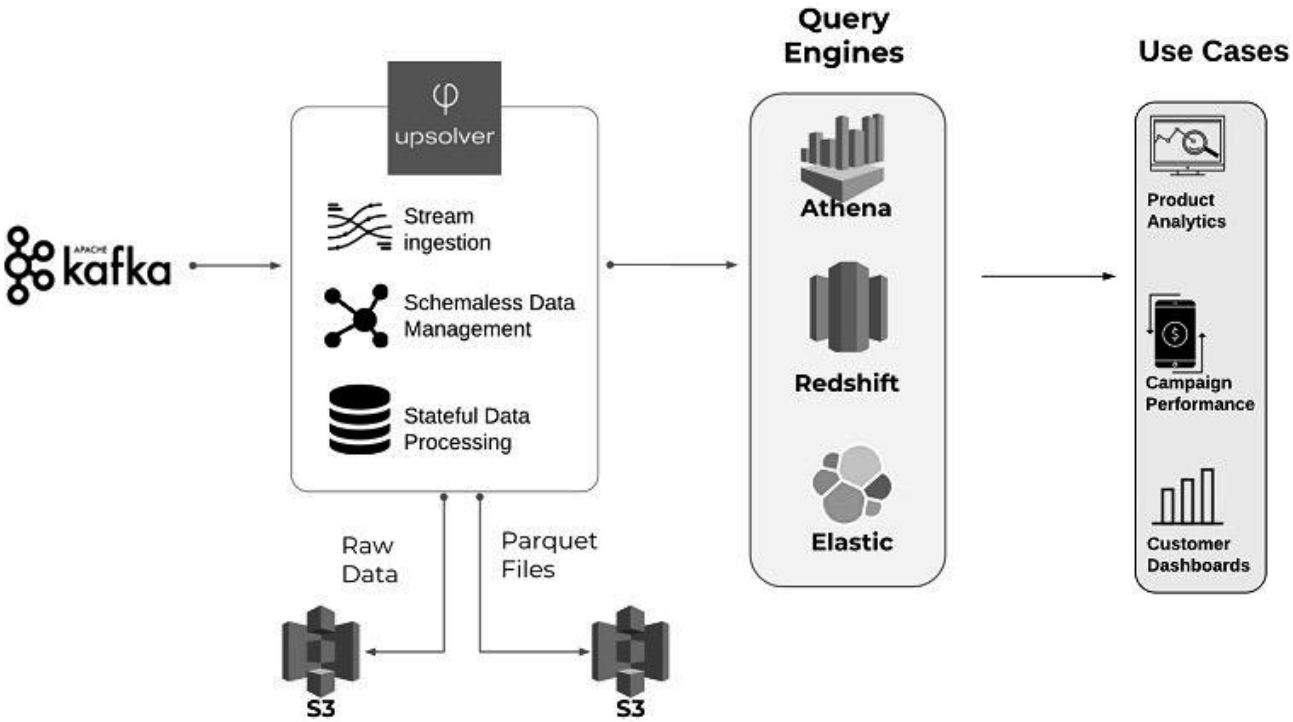


Рис. 2. Архітектурна схема IronSource DataLake[11]

• необхідність додаткових інвестицій у моніторінг системи для створення засобів контролю за збереженням несуперечності;

• подвійна вендорна залежність – компанія залежить і від AWS, як від провайдера хмарних сервісів, і від Upsolver як провайдера платформи інтеграції із постачальниками даних.

SimilarWeb DataLake

SimilarWeb DataLake – це архітектурне рішення компанії для побудови системи Anomaly Detection на Web сторінках.

Виходячи із особливостей використання системи очевидно, що подібне озеро DataLake орієнтоване на короткотривале збереження початкових даних через їх величезну кількість. Система має адаптуватися до неструктурованих даних через те, що компанії необхідно фільтрувати значну кількість веб-сторінок, розроблених різними компаніями, які, відповідно, не стандартизовані.

SimilarWeb також використовує Upsolver для вирішення проблеми потокової обробки потоків даних. На схемі можна побачити, що початкові дані зберігаються в окремому сховищі S3 лише 1 день (термін, необхідний для обробки даних), далі події групуються у логічні групи, виділяються унікальні ключі тощо.

Зі схеми можна також побачити, що SimilarWeb виеористовує AWS Glue Data Catalog для контролю схеми даних. Компанія будує супер-сет моделі даних (конкетенація можливих полів) для подальшого використання у запитах Athena. Athena використовується як система побудови запитів до даних і виявлення даних, що відрізняються від заданих правил або правил розподілу.

Подібна система має обмеження і не може бути повністю автоматизованою, бо різні потоки подій можуть конфліктувати між собою, тому їх потрібно розділяти.

Різні дані одного й того ж потоку також можуть конфліктувати один з одним, тому для реалізації потокової обробки компанія і вирішила використати готовий продукт.

Таким чином, серед значних недоліків цієї архітектури можна також зазначити:

• подвійну залежність від провайдерів послуг (AWS для ресурсів та Upsolver, що повністю контролює логіку потокової обробки);

• відсутність Data Marts і можливості повторно перевикористати вже існуючі обчислення;

• обмеження Athena як провайдера машини SQL.

Опис запропонованої концепції

Запропонована схема має наступний алгоритм роботи:

Потокова передача даних у Pub/Sub – вхідну точку для даних до хмари.

Потокова передача даних із Pub/Sub до Landing Zone.

Паралельний запис кожної події в контролюючу таблицю.

Обраховування зміни даних (дельти) за обраний часовий проміжок.

Оновлення даних по виборці дельти.

Побудова Data Marts.

Підготовка даних для використання кінцевими сервісами-користувачами (ма-

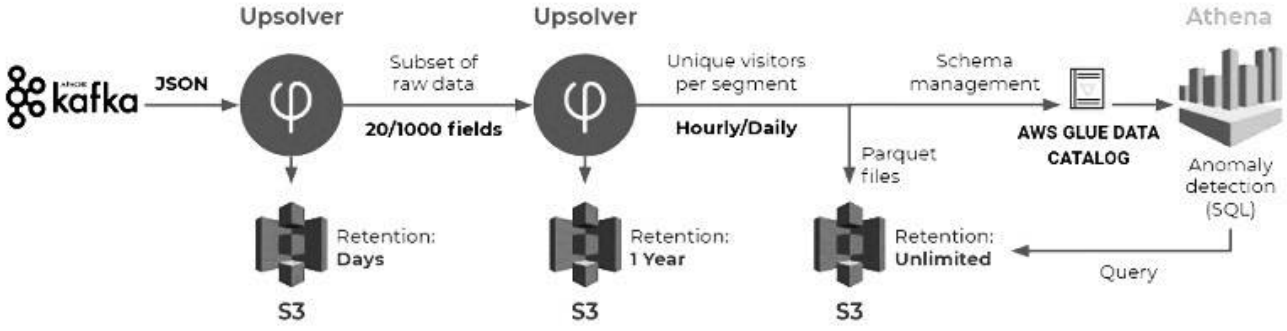


Рис.3. Архітектурна схема SimilarWeb DataLake[12]

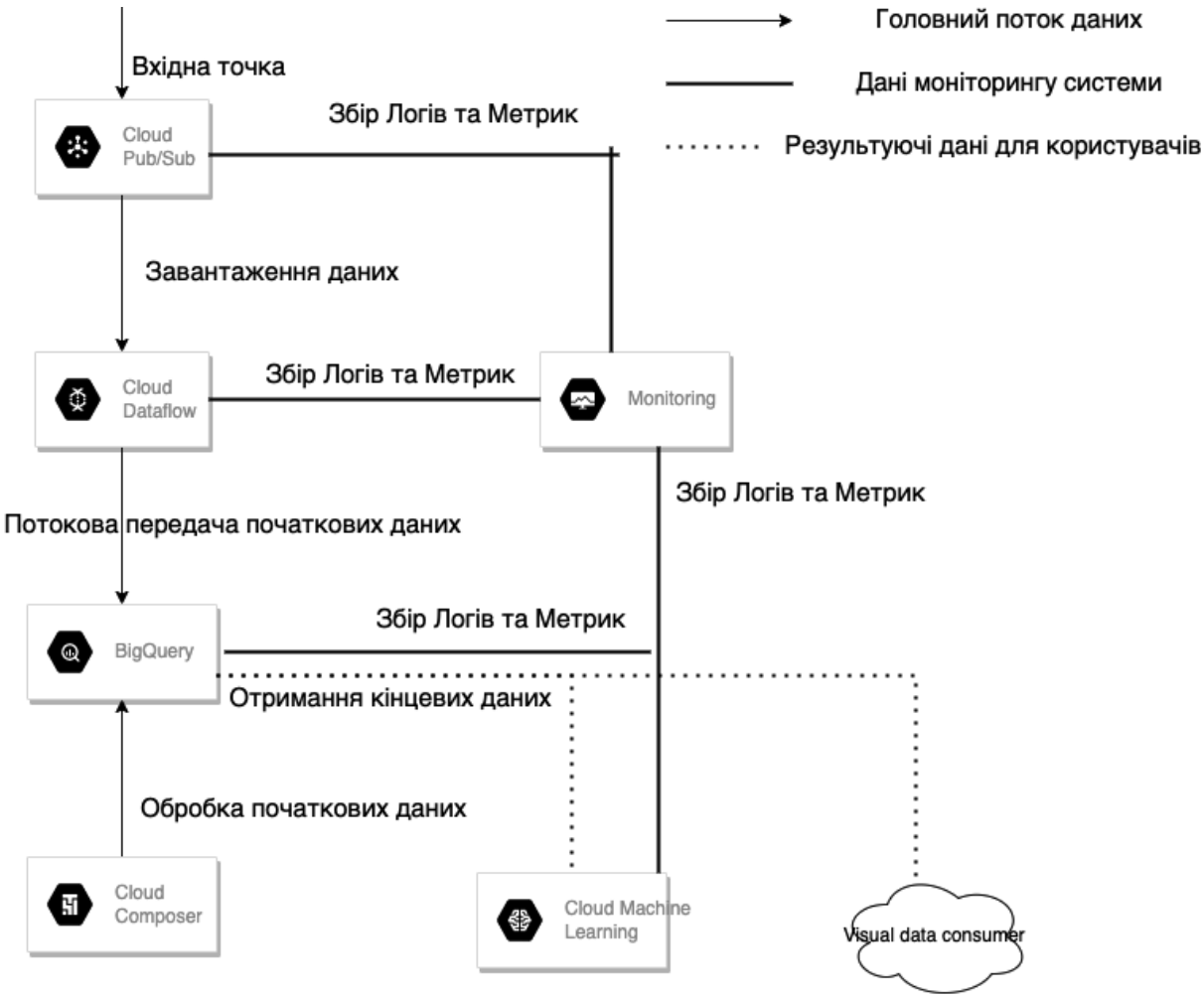


Рис. 4 Запропонована архітектурна схема із використанням GCP

шинне навчання (МЛ), засоби візуалізації, засоби прийняття рішень);

Паралельний моніторинг процесів і ключових метрик.

Засоби безпечного доступу до даних

На сьогодні існує багато загроз, що можуть призвести до несанкціонованого доступу до даних. Такі загрози можуть бути зовнішніми, як - от:

- хакерська атака;
- кібер-атака іншої держави або терористів;
- атака зі сторони іншої компанії або третіх сторін.

Також загрози можуть бути і внутрішніми від джерел (або людей), що навіть не знають про це:

- шахрайство (fraud);
- атака колишнього/діючого працівника з певним рівнем доступу;
- бездіяльність або приховування (працівник міг зробити помилку, що призвела до витоку даних і під страхом наслідків він може приховати факт події);
- ненавмисне заволодіння даними під час розробки чи підтримки системи.

Багато реалізацій аналітичних сховищ мають проблеми із забезпеченням безпеки, особливо в останньому пункті зі списку. Через низький рівень автоматизації такі системи вимагають постійного доступу до даних користувачів для розробників системи.

Запропонована концепція не має такого недоліку, надання доступу до даних dataset відбувається за допомогою Identity and Access Management(IAM), тому лише авторизовані для цієї операції можуть отримати доступ.

Якщо лише частина даних є таємницею, наприклад, userId – не є секретом, бо не є PII, а email користувача – має бути захищеним, то в GCP існує ресурс Data Catalog, який передбачено використовувати під час реалізації концепції для того,

щоб захистити окрему колонку даних. Data Catalog дозволяє створити ярлик (label) на колонку із даними, додати його до групи і надати доступ до групи лише авторизованим користувачам. Тож концепція значно скорочує і визначає коло осіб, що мають доступ до даних користувачів системи.

Для захисту від зовнішніх загроз існує практика не надання прямого доступу до аналітичного сховища стороннім сервісам, навіть тієї ж самої компанії.

Запропонована концепція реалізує цей засіб захисту – доступ до сховища неможливий із зовнішнього середовища, точкою взаємодії із зовнішнім світом є Pub/Sub – для завантаження даних, що передаються потоком Dataflow, який написано інженерами, що розуміють будову сховища і несуть за нього відповідальність. Доступ до Data Mart для користувачів є «тільки для читання», тобто вони не можуть змінити дані, а весь доступ також є авторизованим (і для людей, і для сервісів за допомогою Service Account).

Порівняльний аналіз моделей

Запропонована концепція має значну перевагу у надійності Entry Point. Якщо в SDLF – це сервіс, що зберігає файли (S3), а файли з даними можуть бути великими, то цей стандарт може мати проблеми із незавершеним завантаженням і відповідно втратою даних. На відміну від цього запропонована концепція використовує Pub/Sub [13-16], що здійснює принцип "at least once delivery", який гарантує доставку повідомлення один раз або більше, що виключає можливість втрати даних на цьому кроці. Upsolver використовує для цих цілей Kafka Schema Registry та дає своїм користувачам можливість контролю схеми даних. SimilarWeb додатково використовує AWS Glue Schema Registry для того, щоб мати схему ще й на рівні машини запитів Query Engine, якою в даному випадку є AWS Athena. Kafka є еквівалентною до запропонованого рішення за надійністю, проте значно складнішою у налаштуванні та підтримці (навіть у безсерверній реалізації).

Запропонована концепція має значну перевагу у методі потокової передачі даних над головним конкурентом – SDLF. Він використовує технологію Dataflow [17-18], що працює на шаблонах, які реалізують Apache Beam – провідний стандарт у сфері сховищ даних ETL [17]. Це дозволяє використовувати здобутки бібліотек із відкритим кодом і писати складні потоки із використанням малої кількості самописного коду. Крім того, Dataflow запускається на контрольованому кластері з машин, на відміну від AWS Lambda [19] в SDLF. Це означає, що система краще масштабується і коштує значно дешевше навіть за умови великих і дуже великих даних. Як видно із досвіду IronSource та SimilarWeb багато компаній не бажають мати справу з потоковою обробкою великих даних й купують вже готові сервіси. Однак інженери компанії все одно мають розуміти, як працює їх потокова обробка для того, щоб знати, чи задовольняє реалізація вендора їх бізнес-потреби. Додатково це значно збільшує залежність від провайдера послуг. Запропонована концепція має значну перевагу над архітектурами IronSource та SimilarWeb, тому що використовує бібліоеки з відкритим кодом.

Запропонована концепція має значну перевагу в організації зони Landing над усіма конкурентами за рахунок Dataset в BigQuery. Це дозволяє робити запити до даних мовою SQL, переглядати їх, конвертувати, проводити реконсиляцію (перевірку несуперечності системи) на відміну від інших, де дані фізично зберігаються у файлах, тож пошук конкретного запису фактично неможливий. Серед недоліків цього підходу можна зазначити вищий рівень витрат на сховище, проте якщо брати повну вартість володіння систем TCO, а не окремі витрати на сховище, то різниця буде не такою великою, бо підтримка стає простішою (означає, що треба витрачати менше людино-годин кваліфікованих працівників), немає потреби у окремій системі моніторингу за несуперечністю, результати обробки даних можна перевикористовувати.

Запропонована концепція має значну перевагу у моделі оновлення даних,

адже ця операція виконується за допомогою відкритої бібліотеки Airflow [21], створеної для керування складними ETL-процесами і вже реалізовує дуже багато логіки всередині, яка може бути перевикористана на відміну від реалізації SDLF, де процес обробки даних передано на AWS Lambda, тобто на самописний код.

Ще однією перевагою Airflow є використання кластеру виконувачів на основі Kubernetes. Тобто оплата відбувається за кількістю використаних годин і потужністю інфраструктури кластеру, а не за кількістю викликів. Таким чином запропонована модель обробки даних буде значно дешевшою на великих даних, ніж SDLF.

Вразливість моделей до зміни схеми даних

Однією з найбільших проблем аналітичних баз даних є необхідність адаптуватися до зміни початкових даних, водночас зберігаючи якість кінцевих даних. Очевидним недоліком SDLF є повна відсутність контролю схеми даних, що унеможливлює побудову повноцінного сховища.

Якщо компанія бере на використання SDLF, то підтримка схеми лягає на відповідальність сервісів озера даних Data Lake. Такий підхід до архітектури може бути правильним, проте ця особливість не підкреслена в документації стандарту для того, щоб мати змогу говорити про "дуже швидке розгортання SDLF від розробки до Production". Однак необхідність самостійної інтеграції контролю схем, наприклад, за допомогою Kafka Schema Registry[13-15], забере багато додаткового часу.

На відміну від SDLF, запропонована концепція може працювати у двох режимах.

Перший режим аналогічний до SDLF. Він передбачає передачу відповідальності за схему даних попередньому сервісу, тобто архітектура передбачає, що до сховища доходять лише валідні повідомлення.

Другий режим передбачає використання Pub/Sub Schema Registry, що буде перевіряти схему повідомлення на етапі спроби відправки сервісом постачальника даних. Постачальники, які не будуть відправляти повідомлення у схемі, підтримуваної сховищем, просто не зможуть відправити таке повідомлення й отримають зрозумілу відповідь про помилку.

Такий підхід має значну перевагу через те, що помилки виявляються на першому ж кроці, коли постачальник намагається відправити повідомлення, а не на кінцевому, як в SDLF, коли дані вже у сховищі, проте їх неможливо обробити, через те, що схема змінилася або зовсім невалідна.

Важливо зазначити, що у великих системах, де кількість унікальних типів подій може вимірюватись сотнями і тисячами, в обох концепціях (запропонованому та SDLF стандарті), слід делегувати відповідальність за валідацію схем централізованому сховищу Event Store.

Висновки

У роботі розроблено концепцію організації аналітичного сховища даних, а саме:

- метод взаємодії постачальників даних зі сховищем;
- метод контролю схеми даних;
- метод потокової передачі даних;
- метод зберігання початкових даних;
- метод обробки даних;
- метод надання безпечного доступу до даних.

Розглянуто інші існуючі на ринку стандарти, а саме:

- SDLF – провідний стандарт рекомендований AWS;
- IronSource DL із використанням Upsolver;
- SimilarWeb DL із використанням Upsolver.

Проведено порівняльний аналіз (здебільшого з SDLF, позаяк його реалізація відкрита, а деталі реалізацій приватних компаній є таємницею). Детально оглянуто переваги запропонованої концепції над існуючими. Наведено рекомендації щодо способу інтеграції концепції із застосунками із контролю схеми даних.

Розроблено сервіс потокової передачі даних за допомогою Apache Beam мовою Java.

References

1. *Електронний ресурс [01.02.2022]: https://aws.amazon.com/data-warehouse/*
2. *Електроний ресурс [01.02.2022]: https://cloud.mts.ru/cloud-thinking/blog/data-warehouse/*
3. Електроний ресурс [01.02.2022]: https://azure.microsoft.com/en-us/resources/customer-stories
4. Електроний ресурс [01.02.2022]: https://builtin.com/data-science/company-data-lake-questions
5. Електроний ресурс [01.02.2022]: https://databricks.com/discover/data-lakes/challenges
6. Електроний ресурс [01.02.2022]: https://lingarogroup.com/blog/8-challenges-faced-by-ctos-when-starting-data-lake-projects/
7. *Електроний ресурс [01.*02.2022]: https://www.qlik.com/blog/2020-the-year-cloud-data-warehouses-arrived
8. Електроний ресурс [01.02.2022]: https://www.castordoc.com/blog/cloud-data-warehousing-the-past-present-and-future
9. *Електроний ресурс [01.02.2022]: https://catalog.us-east-1.prod.workshops.aws/v2/workshops/501cb14c-91b3-455c-a2a9-d0a21ce68114/en-US*
10. *Електроний ресурс [01.02.2022]: https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/deploy-and-manage-a-serverless-data-lake-on-the-aws-cloud-by-using-infrastructure-as-code.html*
11. Електроний ресурс [01.02.2022]: https://www.upsolver.com/case-studies/ironsource-how-built-petabyte-scale-data-lake
12. Електроний ресурс [01.02.2022]: https://aws.amazon.com/blogs/big-data/how-similarweb-analyze-hundreds-of-terabytes-of-data-every-month-with-amazon-athena-and-upsolver/
13. *Електроний ресурс [01.02.2022]: https://docs.confluent.io/platform/current/schema-registry/index.html*
14. *Електорний ресурс [01.02.2022]: https://habr.com/ru/company/alfastrah/blog/547092/*
15. *Електроний ресурс [01.*02.2022]: https://www.bigdataschool.ru/blog/kafka-big-data-schema-registry.html
16. *Електроний ресурс [01.*02.2022]: https://cloud.google.com/pubsub/docs/schemas
17. Електроний ресурс [01.02.2022]: https://cloud.google.com/dataflow
18. Електроний ресурс [01.02.2022]: https://habr.com/ru/post/122479/
19. Електроний ресурс [01.02.2022]: https://beam.apache.org/
20. Електроний ресурс [01.02.2022]: https://aws.amazon.com/ru/lambda/
21. Електроний ресурс [01.02.2022]: https://airflow.apache.org/

Отримано: 18.02.2022

***Про авторів*:**

Тюрін Валерій Олександрович,
магістрант Національного
Технічного Університету України
«КПІ імені Ігоря Сікорського».

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії
комп'ютерних обчислень, професор
кафедри інформаційних систем
та технологій Національного
Технічного Університету України
«КПІ імені Ігоря Сікорського».
Кількість наукових публікацій
в українських виданнях – понад 190.
Кількість наукових публікацій
в зарубіжних виданнях – понад 80.
Індекс Хірша – 6.
http://orcid.org/0000-0002-8435-1451

Савчук Олена Володимирівна,
кандидат технічних наук, доцент кафедри
інформаційних систем та технологій
Національного Технічного Університету
України «КПІ імені Ігоря Сікорського».
Кількість наукових публікацій
в українських виданнях – понад 100.
Кількість наукових публікацій
в зарубіжних виданнях – 3.
Індекс Хірша – 2.
http://orcid.org/ 0000-0003-3176-7952

Місце роботи авторів:

Національний технічний університет
України «Київський політехнічний
інститут імені Ігоря Сікорського»,
проспект Перемоги 37
та Інститут програмних систем
НАН України, 03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559
E-mail: tiurinvalery@gmail.com
doroshenkoanatoliy2@gmail.com
elenasavchuk0@gmail.com

УДК517.9:621.325.5:621.382.049.77 http://doi.org/10.15407/pp2022.01.75

M. Kosovets, L. Tovstenko

THE PROBLEM OF DEVELOPING THE ARCHITECTURE OF MODERN COGNITIVE RADAR SYSTEM

The problem of developing the architecture of modern cognitive radar systems using artificial intelligence technologies is considered. The main difference from traditional systems is the use of a trained neural network. The heterogeneous multiprocessor system is rebuilt in the process of solving the problem, providing reliability and solving various types of problems of one class and deep learning of the neural network in real time. This architecture promotes the introduction of cognitive technologies that take into account the requirements for the purpose, the influence of external and internal factors.

Keywords: Perception-Action Cycle, Artificial Intelligence, Signal to Noise Ratio, Active Electronically Scanned Array, Environmental Dynamic Database, Signal to Noise Ratio, Radar Resource Management, multiprocessor.

Introduction

Modern radar systems operate in a wide variety of dynamically changing scenarios: the detection, tracking and classification of very small and slow targets. Such objects as drones, missiles, boats, along with a complex spectrum are important system requirements. Cognitive radars, combining many well-known and new methods, offer a promising solution to these problems. We consider the functional architecture of cognitive radar from the perspective of the user and the manufacturer.

Cognitive radar is an updated technology, the origins of which go back to science - «cybernetics», human-machine interaction, signal processing. The evolution of cognitive radar is aimed at achieving cognition, as in its natural counterparts, such as the radar capabilities of bats and dolphins, or human intellectual decision-making. This article provides an overview and trends in the development of cognitive radar systems.

Organization of cognitive computing

The term «cognitive radar» was first introduced by Dr. Simon Haikin [1], following the ideas of cognitive neurology, which is based on works of cybernetics, artificial neural networks, self-organized learning and solutions of Bayesian theory. Engineering analogues for the implementation of the main cognitive features identified by Faster: memory, attention and intelligence (PAC Perception-Action-Cycle: Cycle-perception-action) have been proposed [2]. In studies of cybernetics, Rasmussen [3], [4] described human behavior in terms of three levels: based on skills, rules and knowledge. He described behavior-based behavior as a subconscious that reflects basic signal processing and generation blocks in a radar system [5]. Rule-based behavior is used in familiar situations. The basis of parallel work is modeling and analysis of previous experience.

Build cognitive radar developers have inspired research in the field of biomimetics. Artificial intelligence is modeled on the basis of observations of living intelligence. Thus, masters of echolocation - bats and dolphins can detect and track very small prey, using complex waveforms that are changed dynamically [6]. Moreover, knowledge of the intelligence of living beings allows us to better understand living nature. It helps to create artificial intelligence, which is superior to «living» and is used in technical systems.

Information in the radar system is perceived by «smart» sensors, i.e. sensors with primary processing and control of the measurement process [7], as well as through network sensors that demonstrate «distrib-

© M. Kosovets, L. Tovstenko, 2022

uted intelligence» with self-monitoring capabilities, automatic solution of changes in their environment [8] , [9]. Cognitive radar has the ability to adapt to transmission in the probing process, imitating human perception as an interactive process where the cognitive entity responds to or changes its behavior as a result of external stimuli.

In traditional radar systems, the flow of information is one-way: the radar interrogates the environment by transmitting a fixed, predetermined pulse signal, regardless of any changes in the environment. Adaptive processing is performed on reception, but the results of such processing do not control any radar function for transmission. An overview of the cognitive directions of radar construction research over the last decade gives an idea of the methods being developed for a wide range of radar applications. Technical problems in the development of cognitive radars are the motivation for further work in this area. Central to these works is the idea of closed-loop data collection, where the dynamic state is interpreted as an adaptive measurement determined by Kalman filtering. This approach allows the antenna array to be adaptively directed in the direction and width of the beam, as well as to place zeros so as to reject any unwanted signals or noise outside the main particle.

A database of problems has been developed that allows comparing methods that use beam control of phased array antennas to optimize tracking, minimizing false alarms [10], [11]. Sometimes several hypotheses and filtering interactions of several tracking models are tested [12-13] to optimize performance: such as signal-to-noise ratio, interference effects, track, and detection threshold.

Common features of increasing adaptability: prediction of adaptive scheduling review time, adaptive choice of detection thresholds (eg, constant false alarm rate detectors) [14], and adaptive interference suppression using adaptive-spatio-temporal processing mode [15] to improve target detection. Adaptive tracking methods vary the measurement time, as well as the signals used to update the trajectory, based on the measurements obtained by the tracker. This feedback is used to control the radar so that frequent measurements are made during an unpredictable or rapid dynamic maneuver, while infrequent measurements are made during predicted periods or steady dynamics.

The simultaneous change of intrapulse signal modulation is studied on the basis of the provided measurements on the tracker. This leads to the choice of optimal signal methods [16-17] and their adaptive extensions [18-19]. Optimization of radar signal in dynamics, to maximize performance according to specific scenarios and tasks, includes the use of some components of radar, such as antenna, radiation pattern (both transmission and reception), time, frequency, coding and polarization. The signals are selected from several classes of signals, such as linear or nonlinear frequency modulation, phase or encoding frequency, and ultra broadband signals. This also includes adapting parameters within the signal class, such as changing the pulse repetition interval, bandwidth, or center frequency [20]. The optimal signal, which maximizes the signal / noise, arises as a solution of the generalized eigenvalue on the waveform [21], developing in the framework of «joint optimization of transmission and reception by the choice of waveform. The approach was used in Bayesian theory of decision making and development, designed to optimize the system by selecting the signal at the transmitter and minimizing interference at the receiver. There are also difficulties in choosing the criteria of optimality and accurate distribution of interference.

According to the IEEE, the definition of «cognitive radar» is a radar that has the ability to learn: «Radar system, which automatically generates a constant perception of the target scene and takes appropriate action. It can use short-term and long-term memory to increase the performance of a given function. Compared to adaptive radar, cognitive radar is trained to adapt operating parameters as well as processing parameters, and can do so over longer periods of time."

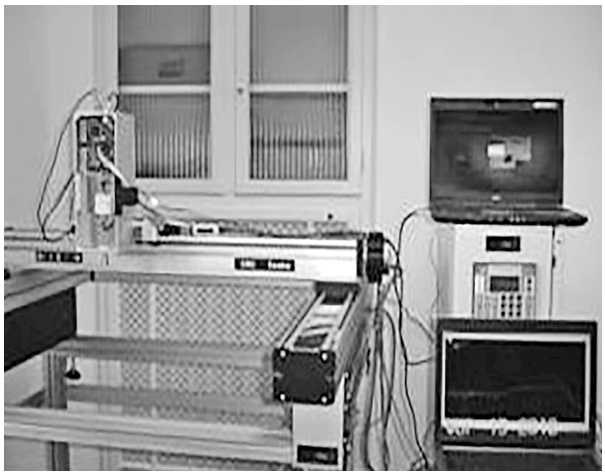


Fig.1. FMCW Radar Imaging Cognitive Ability Modeling Complex with Deep Learning Package.

Cognitive radar differs from traditional active radar due to the following features: development of rules of conduct for self-organization through a process called experiential learning, which is the result of long-term interaction with the environment. According to Charlish, cognitive radar is a radar system that acquires knowledge and understanding of the work environment through online assessment and training from databases that contain contextual information. Cognitive radar uses this knowledge to improve information: search, data processing and management of radar resources. With the development of cognitive radars began a new era in the creation of modern radar systems. The number of publications on the development of cognitive radar architecture with artificial intelligence is increasing avalanche. Artificial intelligence is used in the construction of neural networks, methods of deep learning, signal processing, pattern recognition, classification. It should be noted that elements of cognition in the construction of radars have always been presented (power, pulse width, repetition rate, modulation, etc.). There is also the ideology of the neural network, and accordingly their in-depth training (multiprocessor with restructuring). The explosive growth of the latest developments in radar systems is related to public demand (defense, security, medicine, subsurface sounding, mine search, unmanned aerial vehicles) and the ability to meet them: the use of artificial intelligence technology and the development of a new component base.

Cognitive radar methods use mimic elements of human cognition, such as the cycle of perception-action, deep learning, intelligence and the use of existing knowledge [22]. Cognitive radar vision methods use radar spectrum [23], [24], radiation optimization [25], tracking [26-28], beam control [29], interference reduction [30], network [31], resource management [32] .

To develop a cognitive radar system that adapts in real time, the multiprocessor must be an integral part of the simulation tool. It helps to analyze the behavior of the radar at the simulation stage. The post-process stage consists of two steps. In the first step, a radar sensor and a proven circuit are developed using non-adaptive settings. In the second step, the multiprocessor is tested and configured, replacing the Front-end sensor. Pre-recorded raw datasets that include all radar parameters are optimized. Such datasets use the same measurement for all parameters of environment, target, and trajectory. At this stage of development, the feedback cycle is closed by obtaining an interval of coherent processing of raw data relating to the selected optimal parameters in real time.

The modeling complex consists of an adaptive radar sensor that perceives the environment with optimized radar parameters and a multiprocessor that tracks the target and selects the optimal radar parameters for each new measurement. The sensor consists of an adaptive signal generator, a radar interface, an ADC and a real-time signal converter and a display. The controller consists of a Kalman filter tracker and an optimizer that selects both optimal signals and real-time processing parameters based on the latest measurements. Multiprocessor processing is simulated in Matlab and runs on an Ubuntu Linux PC.

The radar data processing system consists of three modules: FPGA, workstation and graphics processing unit. FPGA provides primary signal processing and hardware management. Its most important role in signal processing is to perform digital down-conversion of the received signal so that the true baseband can be transmitted to the workstation for further processing. Subsequent implementation of the appro-

priate filter on the FPGA can be useful for freeing resources on the graphics processor for its other tasks: filtering, discrete Fourier transform, and processing the constant rate of false alarms (CFAR). The detection task is implemented on a graphics processor. The CPU is an Intel multi-core processor.

LabView National Instruments support control functions. In addition, LabView can be used to write FPGA software. Simulation on FPGA Xilinx, RF-path on on-a-chip is used. The shell is written in Python for the C ++ library. Algorithms are tested on flexible radar equipment. We use digital transceiver systems - such as Universal Software Radio Peripheral (USRP).

In recent years, research on cognitive radar design has been conducted covering a wide range of programs, using many different methods based on previous advances in Bayesian theory, information theory, theoretical solutions, approaches, including fuzzy logic, rule-based systems, metaheuristic algorithms and Markov solutions. processes, dynamic programming, optimization and game theory.

Future systems are learning the ability to predict the behavior of radars in the operational environment and to adapt its transmission in the available spectrum. Radar cognition in this case is based on two main concepts: spectrum probing and spectrum distribution. The sounding spectrum is aimed at recognizing the frequency used by other systems and occupying the same spectrum in real time.

System performance is measured in terms of standard metrics such as target detection probability and false alarms, root mean square error in tracking systems, and classification accuracy in automated target recognition systems - cognitive systems require additional metrics that quantify performance gains and achievement use of system resources. Two key issues for cognitive radar research are the development of assessment and assessment tools, as well as experimental testing of the methodology.

A related but unique problem with the cognitive design of radar is experimental testing, as the shape of the transmitted signal and the settings are adapted during operation. With more sophisticated modeling, new development and qualification processes can be developed, including software testing that will help test cognitive radars.

Cognitive radars are evaluated through simulations, or using pre-recorded data. The infrastructure of testing, calibration and debugging tools is being developed in parallel. For example, SPC Quantor has developed real-time tests for cognitive radars. Reliability of modeling and computational errors is an important issue that should be investigated [33].

Radars differ in their qualitative and quantitative parameters. A typical approach is to determine the number of worst cases and make them work in the worst cases. This is true for non-cognitive radar systems that do not change the configuration depending on the current environment, because a single radar configuration is used. Cognitive radars change their configuration, rebuilding the neural network and learning or self-learning to solve various problems within certain limits. Moreover, with the development of neural network design tools, cognitive radars will have better characteristics and lower design costs and the ability to self-improvement.

The power of the transmitter should not exceed the limits imposed by regulatory requirements, because there is unwanted radiation due to the nonlinearity of the transmitter and a sharp increase and decrease in radar pulses [34]. Especially in cognitive systems, dynamic reconfiguration of the transmission spectrum is not always easy to implement and can lead to out-of-band transmissions, which cause a slight spectral expansion outside the designated radar band.

Cognitive radar architecture

The cognitive radar architecture is built through the extension of the perception-action cycle by introducing an evaluation process that forms a perception-evaluation-action cycle (PEAC). The purpose of this additional step is to emphasize the assessment of the currently perceived situation supported by artificial intelligence. It is done regarding the purpose of the sensor and depends on the purpose certain perception results will lead

to very different actions, such as observation, where the overall picture is important. When observed, all identified targets will receive an equal share of available resources.

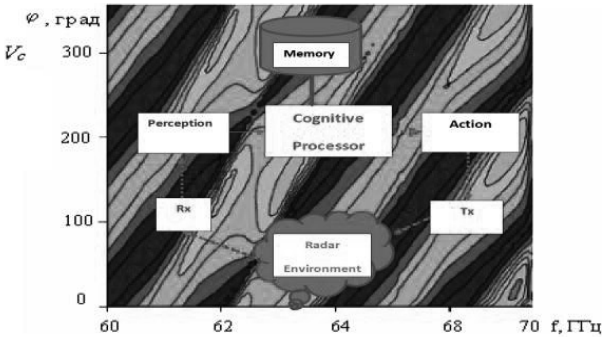


Fig.2. Functional diagram of the cognitive radar.

The radar sensor, which operates in PEAC, consists of a programmable sensor that provides data processing and data evaluation, and management of resources generated depending on recently received environmental information. In future systems, much of the intelligence will be located in a multiplatform cloud. Cognitive radar responds intelligently to real-time scenario variations.

A software-defined sensor is a system that performs radar measurements, ie emits, receives and processes electromagnetic signals in accordance with the requirements of the sensor in order to obtain new information in its environment. A software-defined sensor requires hardware capabilities (eg, instantaneous bandwidth, operating bandwidth, waveform, polarization, etc.) to meet resource management requirements.

Radio compatibility is achieved by methods relating to the emission of radar signal or signal processing. Methods of reducing interference from other radio frequency systems are achieved by dividing in time, frequency, space or signal modulation. Consider the main measures that allow coexistence: waveform, illumination in radar images, adaptive zeroing of interference, frequency adaptation, dynamic adaptation of the search circuit and the level of radiation power, detection of interfering samples in the receiving signal, suppression of interfered samples in the radar signal [35].

A Doppler shift around a target appears if the target contains moving, vibrating, or rotating parts and can be observed externally. For example, the wings of birds, the wheels of cars, the arms and legs of people walking, as well as the rotors of helicopters and tank tracks have a unique Doppler spectrum, which is observed using a radar system with a fairly high Doppler resolution. The spectrum of Doppler shift strongly depends on such parameters as the angle of illumination, the absolute velocity of the target and the composition of the underlying surface.

To adjust the detection thresholds according to the actual background, it is necessary to determine and assess the level of interference. This assessment can be performed on a one-time basis or over a long period of time by studying the characteristics of the obstacles. The mapping features can be characterized by its spatial composition, amplitude statistics and Doppler spectrum. This allows you to reliably adjust the detection thresholds according to the characteristics of the interference, while maintaining a low level of false positives and providing the ability to detect targets against the background of interference.

Multi-beam radiation on the sea surface is characteristic of marine radars. The imposition of a direct and reflected path on the sea surface leads to the appearance of zones with attenuation and loss of target detection, as well as to errors in altitude measurement. By detecting fading situations, tracking can be more resistant to detection errors by changing the transmission frequency and thus avoiding the fading situation.

Active grating and electronic scanning antennas, which are the latest in modern radar systems, provide great flexibility in the direction of the beam and waveforms.

The various actions that require resources from the system are called tasks. Each task is an implementation of the radar function that the sensor is capable of. Examples of such tasks: search tasks, tracking tasks, classification tasks, visualization tasks, environmental detection tasks, externally assigned search and tracking tasks ordered by a higher system.

QoS resource management techniques use quality measures to optimize overall sys-

tem performance for this metric and available resources. Therefore, this approach requires a good understanding of the quality measures used. Especially when a large number of different tasks are used, which will certainly be the case in future cognitive radars, it is necessary to find a strong balance between individual tasks and mission needs. The advantage of the QoS approach is that even in dense scenarios, the available resource is distributed among the tasks still to maximize system performance, and no predetermined priority, which may or may not be applied in the current situation, should be used to resolve conflicts. However, adaptive rules as well as the QoS approach require more environmental information to adapt the waveform to the evolving situation, taking into account the various influences implemented in the model used to assess the expected performance of the system. In addition, for QoS it is necessary to keep a list of all active tasks of the radar sensor.

The hardware capabilities needed to take advantage of modern resource management capabilities are high if you use the full potential of algorithms. In this case, you need a fully flexible interface that allows you to configure all available parameters (such as waveform, parameters, and viewing directions) within the physical boundary of the external interface. However, to speed up the process of optimizing resource management, the available degrees of freedom can be limited in advance, for example, by limiting the repetition rates of the selected pulses. If the limit is chosen adequately, the decrease in productivity is insignificant.

Deep learning of the cognitive radar neural network

Probably, today only the lazy are not engaged in machine learning. But when we look at cognitive radar software, we are talking about deep learning. This is when the feedback covers the entire radar.

The backpropagation algorithm is an extension of the perception of multilayer neural networks. Thus, the backpropagation algorithm uses three or more levels of processing units (neurons). In a typical 3-tier network architecture for a backpropagation algorithm, the leftmost layer of ones is the input layer that receives the input data. Later, this is a hidden layer, where processing blocks are linked to the layers before and after it. The rightmost layer is the output layer. The levels are fully interconnected, which means that each processing unit is connected to each unit at the previous level and at the next level. However, the units are not linked to other units in the same layer. Backpropagation networks are not fully interconnected, which means that any number of hidden layers can be used. [31].

Traditional event detection in cognitive imaging radar is based on batch or offline algorithms: it is assumed that there is one event in each radar information stream. The stream is usually processed using a preprocessing algorithm that requires a huge amount of computation. Neural networks can easily cope with such tasks with the appropriate deep learning. This is an analogue of information processing tasks “on the fly” as they become available.

Neural networks are also an effective method for diagnosing faults based on nonlinear mapping of input and output data, parallel processing and a high degree of self-organization and self-learning ability [36]. In the structure of closed-loop neural networks the only suitable connections are between the outputs of each level and the input of the next level [37]. A backpropagation neural network is one known method for creating a trained machine or system that can provide a final classification decision through a series of learning processes. It can be developed using the tools provided in MATLAB, but sometimes this leads to different detection and recognition accuracy of objects for each experiment [30–31].

We achieve troubleshooting by rebuilding the computational resource of the neural network. Moreover, a quick response occurs by changing the course of the computing process and in case of failures, readjustment of the network with a change in its resource. It is possible to draw an analogy with a living intellect, where homeostasis is provided at the hormonal level and a quick response to changes in the external environment by nervous signaling.

We always willingly or unwillingly use bionic models. Now it has resulted in a separate science of imitation of nature - biomimetics. Creating a model in biomimetics is half the battle. To solve a specific practical problem, it is necessary not only to check the presence of the model properties of interest to practice, but also to develop methods for calculating the predetermined technical characteristics of the device, to develop synthesis methods that ensure the achievement of the indicators required in the problem.

And therefore, many bionic models, before they receive technical implementation, begin their life on a computer. A mathematical description of the model is constructed. Based on it, a computer program is compiled - a bionic model. On such a computer model, various parameters can be processed in a short time and design flaws can be eliminated.

Traditionally, deep learning algorithms update the weight of the network, while the architecture of the network is selected manually using the trial-and-error method. This study proposes two new approaches that automatically update the structure of the network, as well as studying its weight. The novelty of this approach is parameterization, where depth or additional complexity is constantly encapsulated in the space of parameters that give additional complexity.

Deep learning includes several levels of nonlinear information processing. This allows us to study architectures that implement functions through repetitive compositions of simpler functions, thereby exploring levels of abstraction with the best generalization and representation.

Although in-depth training is useful, keeping multiple layers can be problematic. First, when more layers, weight, space, and computational complexity are higher; second, when there are more free parameters, there is a higher risk of retraining; third, if the network is deep, there is the problem of disappearing gradients when the error spreads over many layers.

There have been many approaches to optimizing the network architecture - from early incremental methods of bringing hidden modules one after the other (or starting from a large network and reducing it) to more sophisticated modern approaches such as evolutionary algorithms or reinforced learning and stimulus style techniques. The purpose of the study is to study network architecture based on data. The main difference is that instead of searching in discrete space for all architectures that have parameterized models in such a way that the very notion of complexity or depth is itself continuous, making the model differentiated from beginning to end.

Two methods are proposed for constructing and studying the structure of a deep neural network, where the complexity of the network at the level of a hidden block or layer is encoded by continuous parameters. These parameters are adjusted together with the network weights during the gradient descent, which implies a slight change in the structure of the network together with the network weights. The first method in tunnel networks associated with each hidden block is a continuous parameter. If this parameter is not active, the block simply copies its input to its output to bypass nonlinearity, effectively increasing the depth of the network. In the second method, the perceptron has parameters associated with each layer, indicating whether further nonlinear processing is required. We start with one layer first, and when training with a gradient descent, when necessary, this parameter can become active, which causes the creation of another complete layer, increasing the depth of the network.

Experiments on synthetic double-helix data like tunnel networks and novice perceptrons can be adapted to different sizes for different complexity of problems using the same set of hyperparameters, adapting the number of units for the tunnel networks and the number of layers for the initial perceptrons. With regard to real problems of recognition of numbers and images, we observe that tunnel networks achieve better performance, providing a better regularized model and using fewer parameters compared to backbone networks. Also, novice perceptrons showed comparable or better performance. Compared to tunnel networks,

novice perceptrons appear to grow larger and shrink less. By setting the learning rate in descending order, it is observed that different layers grow at different rates and are used in different ways. Combined with regularization, this allows tunnel networks to keep some of the unused upper layers linear, thus effectively removing them from the network at the end.

Deep learning is an AI function that mimics the workings of the human brain in such a way that it processes data and creates patterns for use in decision making [35]. Cognition is a fundamental feature of natural intelligence. Sensory cognitive networks provide new technological support to dramatically increase the quantity and quality of information that can be collected and transmitted in complex adaptive systems. Their application can significantly increase the level of intelligence in the design and implementation of the system to the levels at which the effects of cognition will begin to manifest themselves. Cognitive abilities can be thought of as a shared sensory network. The detection system learns to detect changes not only in signal levels, but also in the shape and parameters of the sensor signal, which is a more difficult task. The architecture can significantly reduce resource consumption without sacrificing change detection performance. Experiments prove that a neural network-based change detection system is feasible for developing sensor network applications and can be successfully implemented on available technology platforms.

Designing cognitive radars has several stages. At the first stage we develop user requirements. The second is the formalization of user requirements. Next, we develop a model of radar operation, check the receipt of the declared quality characteristics of the radar. In the fourth stage, we conduct in-depth training of the generated neural network with an inverse loop, for which the radar is calibrated. Based on the calibration results, the development of the developing cognitive radar system is adjusted. Consider in more detail the calibration of 3D-Imaging radar, developed and manufactured in SPC «Quantor» on the example of obtaining a 3D image of the internal structure of the multilayer material.

The possibility calibration of is studied in the exploring of material properties to the example of multilayer structure, depending on the distance between the sample and the antenna using an absorber. The results of preliminary studies indicate the possibility of measuring the thickness of the material. On the calibration, a small metal plate and several measurement cycles for averaging the noise were used. It is shown that the accuracy of measurements is influenced by the width of the radiation pattern, the number of measurement cycles at one point, the accuracy of positioning and moving the head during the measurements, and the time interval between the calibrations.

We have developed algorithms and have obtained the required accuracy. We will try to test the radar system, having previously calibrated it.

Before carrying out the measurements, we set:

1. The horn and the sample close the absorber to reduce the reflections;
2. We measure the signal without a sample;
3. We place the sample (5-10) mm and begin to measure;
4. Very carefully, a thin conductive film is pasted from above and measurements are taken;
5. Very gently flip back with a conducting medium and measure.

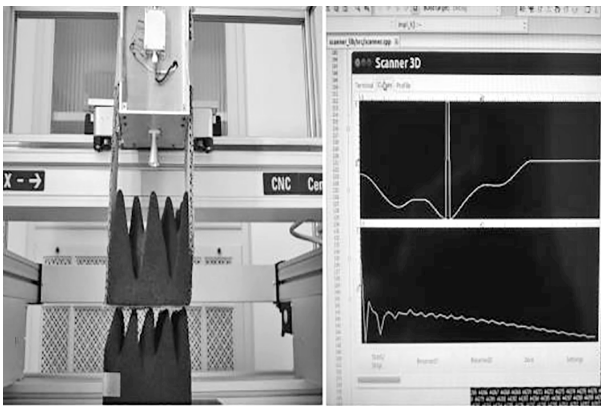


Fig.3. Implementation of 3D scanning of small objects by cognitive FMCW 3D-Imaging Radar terahertz range.

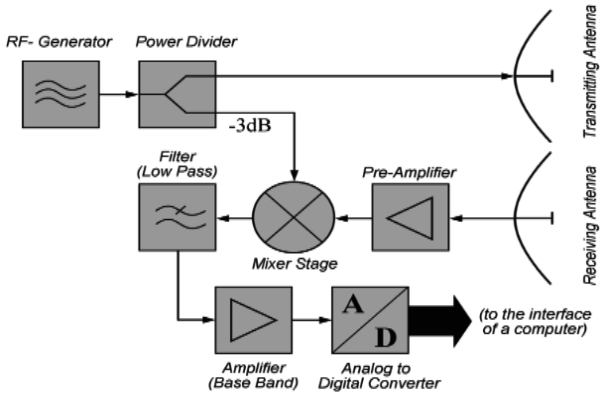


Fig.4. Functional diagram of 3D Imaging FMCW Terahertz Radar.

If all done carefully, there should be a shift of even a fraction of a millimeter. Scan must be disabled. We will make a point of 50 measurements.

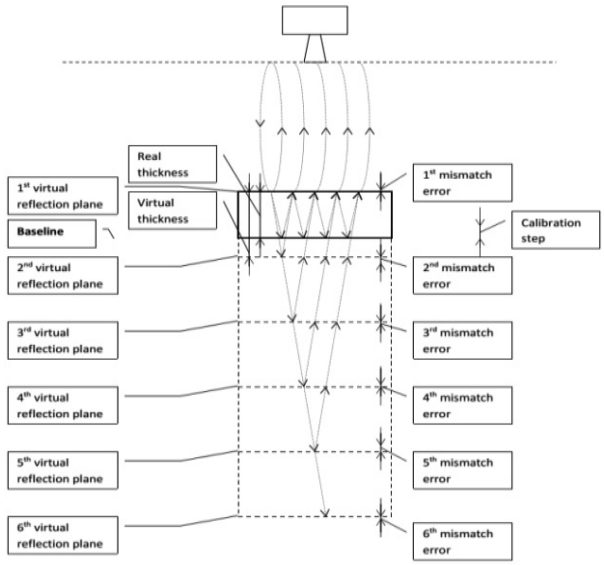


Fig.5. Propagation of radar radiation in a multilayer material.

1. Different materials have a different permittivity and different velocity of phase of electromagnetic wave. This gives that real thickness between upper and lower plane of samples is equivalent to virtual thickness between real upper and virtual lower plane in air. We can estimate equivalent virtual thickness between metal planes in air and then correct result for real material.

2. In real materials we have a multiple reflection. This gives several spectral lines for one thickness of the sample.

3. Reflections from virtual metal planes do not fully correspond to reflections from real metal planes during calibration. There are numerous errors of discrepancy between virtual planes and the nearest calibration levels. This gives multiple errors in the spectral lines and creates some difficulties in estimating the thickness.

4. Some calibration levels must be presented lower than baseline to estimate positions of virtual metal planes.

5. We try to fix existing mathematic problems and to get a mathematical tool for universal measurement device.

6. We check the additional measurement configuration. For calibration, we use a special sample with a higher accuracy – Plexiglas.

As a result of the measurement cycle, a frequency dependence of the attenuation in the microwave channel $D(f) = U_{\text{ref}}(f)/U_{\text{inc}}(f)$ is obtained.

Unknown parameters of the dielectric structure are determined by procedure of global minimization of discrepancy between the measured attenuation in channel $D(f)$ and one calculated theoretically $D_{\text{th}}(f, \text{p})$

$$F(\mathbf{p}) = \sum_f |D(f) - D_{th}(f, \mathbf{p})|^2 .$$

Here $D_{\text{th}}(f, \text{p})$ is defined according to the formula

$$D_{th} = \left| k_0 + k_1 \frac{V - V_c}{(1 - k_3 V)(1 - k_3 V_c) - k_2 V V_c} \right|^2 ,$$

and $k_0(f),.., k_3(f)$ are complex coefficients, which are determined experimentally using reference samples and describe properties of the microwave channel; f is the frequency of sounding waves; $V_C(f)$ is the complex reflection coefficient (CRC) of the reference arm 3; $V(f,p)$ is a theoretically calculated CRC of the dielectric structure, which depends on a vector of the structure parameters p (thickness of layers and electrical parameters of materials).

We consider that in free space extends a plane electromagnetic wave and normally incident on the infinite (M-1)-layer medium with flat boundaries. The CRC $V(f, p)$ is related of the CRC of the structure in free space $V_S(f, p)$ through the scattering matrix of the antenna **S**, which is determined experimentally:

$$V = S_{11} + \frac{S_{21} V_S}{1 - S_{22} V_S} .$$

The CRC of the structure in free space depends on the thickness and electrophysical parameters of structure layers:

$$V_S=V_S(f,h_1,..,h_{M-1},\varepsilon_1,..,\varepsilon_M,\mathrm{tg}\delta_1,...,\mathrm{tg}\delta_M),$$

where h_m, ε_m, $\mathrm{tg}\delta_m$ is thickness, permittivity and loss tangent of m-th layer. The CRC of the plane wave from dielectric plane-layered medium $V_S(f,p)$ is determined by the known formulas:

$$V_S = \frac{W_0 - Y_1}{W_0 + Y_1},$$

$$Y_m = W_m \frac{Y_{m+1}(\exp q_m + 1) + W_m(\exp q_m - 1)}{W_m(\exp q_m + 1) + Y_{m+1}(\exp q_m - 1)},$$

$$Y_M = W_M,$$

$$W_m = \sqrt{\varepsilon'_m / \mu_0};$$

$$q_m = 2j \cdot 2\pi f \cdot h_m \sqrt{\varepsilon'_m \mu_0 (1 - \sin^2 \theta)};$$

$$\varepsilon'_m = \varepsilon_0 \varepsilon_m (1 - j \cdot \mathrm{tg}\delta_m),$$

where ε_0, is permittivity and μ_0 is the permeability of free space.

During the setup process, we do not need to change the distance between the signal and the base line, but we need to will move the device and perform a calibration at the center of each step. This calibration process is simpler and can be performed in automatic mode without an additional table with a micrometer.

Conclusions

This article provides a summary of the development of modern radar systems. It is shown that with the development of Artificial Intelligence technologies, modern radars use deep learning neural networks, as a result, radars have become cognitive. There is no alternative to satisfy the consumer in terms of quality indicators using old technologies. Scientific, technological, information base is ready for such challenges. The need for modern radars is also huge: medicine, security, defense, the Internet of things, and others. Scenarios are becoming more complex and require creative solutions. Cognitive radar is one potential solution that has long been discussed in the literature.

It has been shown that cognitive radars can coexist in a congested spectrum, including with random and intentional interference, and be invisible. Cognitive radar systems can adapt to a changing environment using internal and external sources of information. It is possible to control the resources of the radar, and therefore, the radars are inherently fault-tolerant.

References

1. S. Haykin, “Adaptive radar: Evolution to cognitive radar,” in Proc. IEEE Int. Symp. Phased Array Syst. Technol., Boston, MA, USA, Oct. 2003, p. 613.
2. S. Haykin, Y. Xue, and P. Setoodeh, “Cognitive radar: Step toward bridging the gap between neuroscience and engineering,” Proc. IEEE, vol. 100, no. 11, pp. 3102–3130, Nov. 2012.
3. J. Rasmussen, “Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models,” IEEE Trans. Syst., Man, Cybern., vol. SMC-13, no. 3, pp. 257–266, May 1983.
4. J. Rasmussen, Information Processing and Human Machine Interaction: An Approach to Cognitive Engineering. New York, NY, USA: Elsevier Science, 1983.
5. S. Bruggenwirth, “Design and implementation of a three-layer cognitive radar architecture,” in Proc. 50th Asilomar Conf. Signals, Syst. Comput., Nov. 2016, pp. 929–933.
6. A. Balleri, H. Griffiths, and C. Baker, Biologically Inspired Radar and Sonar: Lessons from Nature. IET, 2017.
7. S. Middelhoek, J. B. Angell, and D. J. W. Noorlag, “Microprocessors get integrated sensors: Sensing devices and signal processing built into one silicon chip portend a new class of smart sensors,” IEEE Spectr., vol. 17, no. 2, pp. 42–46, Feb. 1980.
8. L. Reznik, G. Von Pless, and T. Al Karim, “Distributed neural networks for signal change detection: On the way to cognition in sensor networks,” IEEE Sensors J., vol. 11, no. 3, pp. 791–798, Mar. 2011.
9. R. V. Kulkarni, A. Forster, and G. K. Venayagamoorthy, “Computational intelligence in wireless sensor networks: A survey,” IEEE Commun. Surv. Tut., vol. 13, no. 1, pp. 68–96, Jan.–Mar. 2011.

10. W. D. Blair, G. A. Watson, and S. A. Hoffman, “Benchmark problem for beam pointing control of phased array radar against maneuvering targets,” in Proc. Amer. Control Conf., vol. 2, pp. 2071–2075, Jun. 1994.
11. W. D. Blair, G. A. Watson, T. Kirubarajan, and Y. Bar-Shalom, “Benchmark for radar allocation and tracking in ECM,” IEEE Trans. Aerosp. Electron. Syst., vol. 34, no. 4, pp. 1097–1114, Oct. 1998.
12. G. van Keuk and S. S. Blackman, “On phased-array radar tracking and parameter control,” IEEE Trans. Aerosp. Electron. Syst., vol. 29, no. 1, pp. 186–194, Jan. 1993.
13. S. S. Blackman, R. J. Dempster, M. T. Busch, and R. F. Popoli, “IMM/MHT solution to radar benchmark tracking problem,” IEEE Trans. Aerosp. Electron. Syst., vol. 35, no. 2, pp. 730–738, Apr. 1999.
14. T. Kirubarajan, Y. Bar-Shalom, W. D. Blair, and G. A. Watson, “IMMPDAF for radar management and tracking benchmark with ECM,” IEEE Trans. Aerosp. Electron. Syst., vol. 34, no. 4, pp. 1115–1134, Oct. 1998.
15. J. Ward, Space-Time Adaptive Processing for Airborne Radar. Cambridge, MA: MIT Lincoln Laboratory, 1994.
16. W. K. Stafford, “Real time control of a multifunction electronically scanned adaptive radar (mesar),” in Proc. IEE Colloq. Real-Time Manage. Adaptive Radar Syst., Jun. 1990, pp. 7/1–7/5.
17. S. Haykin, “Radar vision,” in Proc. IEEE Int. Conf. Radar, May 1990, pp. 585–588.
18. S. J. Anderson, “Adaptive remote sensing with HF skywave radar,” IEE Proc. F—Radar Signal Process., vol. 139, no. 2, pp. 182–192, Apr. 1992.
19. D. J. Kershaw and R. J. Evans, “Optimal waveform selection for tracking systems,” IEEE Trans. Inf. Theory, vol. 40, no. 5, pp. 1536–1550, Sep. 1994.
20. D. J. Kershaw and R. J. Evans, “Waveform selective probabilistic data association,” IEEE Trans. Aerosp. Elect. Syst., vol. 33, no. 4, pp. 1180–1188, Oct. 1997.
21. B. F. La Scala, W. Moran, and R. J. Evans, “Optimal adaptive waveform selection for target detection,” in Proc. IEEE. Int. Conf. Radar (Cat. 03EX695), Sep. 2003, pp. 492–496.
22. J. Fuster, Cortex and Mind: Unifying Cognition. New York, NY, USA: Oxford Univ. Press, 2010
23. P. Stinco, M. S. Greco, and F. Gini, “Spectrum sensing and sharing for cognitive radars,” IET Radar, Sonar Navigation, vol. 10, no. 3, pp. 595–602, 2016.
24. A. Aubry, V. Carotenuto, A. D. Maio, and S. Iommelli, “Cognitive radar waveform design for spectral compatibility,” in Proc. Sensor Signal Process. Defense, 2016, pp. 1–5.
25. N. A. Goodman, “Closed-loop radar with adaptively matched waveforms,” in Proc. Int. Conf. Electromagn. Adv. Appl., 2007, pp. 468–471.
26. K. L. Bell, C. J. Baker, G. E. Smith, J. T. Johnson, and M. Rangaswamy, “Fully adaptive radar for target tracking part I: Single target tracking,” in Proc. IEEE Radar Conf., 2014, pp. 0303–0308.
27. J. M. Christiansen, K. E. Olsen, and G. E. Smith, “Fully adaptive radar for track update control,” in Proc. IEEE Radar Conf., 2018, pp. 0400–0404.
28. L. Ubeda-Medina and J. Grajal, “Multiple target tracking in the fully adaptive radar framework,” in Proc. IEEE Statist. Signal Process. Workshop, 2016, pp. 1–5.
29. R. Romero and N. Goodman, “Cognitive radar network: Cooperative adaptive beams teering for integrated search and track application,” IEEE Trans. Aerosp. Electron. Syst., vol. 49, no. 2, pp. 915–931, Apr. 2013.
30. R. Oechslin, P. Wellig, S. Hinrichsen, S. Wieland, U. Aulenbacher, and K. Rech, “Cognitive radar parameter optimization in a congested spectrum environment,” in Proc. IEEE Radar Conf., 2018, pp. 0218–0223.
31. F. Smits, A. Huizing, W. van Rossum, and P. Hiemstra, “A cognitive radar network: Architecture and application to multiplatform radar management,” in Proc. 5th Eur. Radar Conf., 2008, pp. 312–315.
32. A. Charlish and F. Hoffmann, “Cognitive radar management,” in Novel Radar Techniques and Applications: Waveform Diversity and Cognitive Radar and Target Tracking and Data Fusion 2, London U.K.: Institution of Engineering and Technology, 2017.
33. L. Ubeda-Medina, A F. Garca-Fernndez, and J. Grajal, “Robust sensor parameter selection in fully adaptive radar using a sigma-point gaussian approximation,” in Proc. IEEE Radar Conf., Apr. 2018, pp. 0263–0268.

34. R. Turyn, “On barker codes of even length,” Proc. IEEE, vol. 51, no. 1, pp. 230–231, Sep. 1963. [34] D. DeLong and E. Hofstetter, “On the design of optimum radar waveforms for clutter rejection,” IEEE Trans. Inf. Theory, vol. 13, no. 3, pp. 454–463, Jul. 1967.
35. K. L. Bell, C. J. Baker, G. E. Smith, J. T. Johnson, and M. Rangaswamy, “Fully adaptive radar for target tracking part I: Single target tracking,” in Proc. IEEE Radar Conf., 2014, pp. 0303–0308.
36. J. M. Christiansen, K. E. Olsen, and G. E. Smith, “Fully adaptive radar for track update control,” in Proc. IEEE Radar Conf., 2018, pp. 0400–0404.
37. L. Ubeda-Medina and J. Grajal, “Multiple target tracking in the fully adaptive radar framework,” in Proc. IEEE Statist. Signal Process. Workshop, 2016, pp. 1–5.
38. R. Rovol. 49, no. 2, pp. 915–931, Apr. 2013.

Received: 05.02.2022

About the authors:

Mykola Kosovets
Leading Constructor,
Number of scientific publications
in Ukrainian publications -56
Number of scientific publications
in foreign publications -17
Index Girsh - 5
https://orcid.org/0000-0001- 8443-7805
Scopus Author ID: 5644007500

Lilia Tovstenko
Leading Software Engineer
Number of scientific publications
in Ukrainian publications -24
Number of scientific publications
in foreign lands -8
Index Girsh -7
https://orcid.org/0000-0002-3348-6065
Scopus Author ID: 56439972800

Place of work:

Mykola Kosovets
SPE “Quantor”, Chief
03057, c. Kyiv-57, str. E. Potye, 8-A
Ph.: (380)66-2554143
E-mail: quantor.nik@gmail.com

Lilia Tovstenko
Institute of Cybernetics of Glouchkov
National Academy of the National Academy
Sciences Ukraine
03187, Kyiv-187,
Academician Glouchkov Avenue, 40.
Ph.: (380)67-7774010
E-mail: 115lili@incyb.kiev.ua

УДК 004.89 https://doi.org/10.15407/pp.2022.01.87

О.С. Комарський, А.Ю. Дорошенко

МОДЕЛЬ РЕКУРЕНТНОЇ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ГЕНЕРАЦІЇ МУЗИКИ

У роботі розглядається можливість генерації музичних композицій, використовуючи рекурентні нейронні мережі. Запропоновано та розглянуто два методи генерації музичних творів – на рівні нот та на рівні акордів. Проведено дослідження обох методів та визначено їх переваги та недоліки. Далі для розробки обрано метод генерації на рівні нот, для якого детально описано процес пошуку та обробки даних для навчання генерації музики за допомогою рекурентної нейронної мережі. Це дозволяє автоматизувати генерацію музичних творів без втручання людини. Для побудованої моделі виконано програмну реалізацію запропонованого рішення, проведені експерименти та їх верифікація за участю фокус-групи людей щодо визначення авторства створеної музики – людина чи комп'ютер.
Ключові слова: генерація музики, машинне навчання, рекурентні нейронні мережі, LSTM, RNN

Вступ

Машинне навчання покращує багато аспектів нашого повсякденного життя, як очевидних, так і непомітних. Штучний інтелект має велику вагу у таких речах, як системи рекомендацій в різних медіа-сервісах, розумні будинки, комп'ютерний зір, системи виявлення підозрілих дій тощо. І, незважаючи на те, що досі триває дискусія навколо нейронних мереж як чорної скриньки – існує величезний потенціал для їх використання в різних сферах людської діяльності, включно із медициною, віртуальними помічниками, електронною комерцією та сферою фінансів.

Водночас не менш важливий та цікавий напрям, в якому машинне навчання може відіграти свою роль – це творчість. Комп'ютерна творчість є дійсно захоплюючою сферою, в якій штучний інтелект перетинається з мистецтвом, а метою є розроблення систем, які здатні моделювати, вдосконалювати або розуміти творчість людини.

Генерація музики є найбільш абстрактним напрямком творчої діяльності, оскільки на відміну від написання книжок, картин або віршів, музика не прив'язана до контексту оточуючого нас світу. Водночас запис музики може бути представлений у вигляді дуже простого символьного формату даних без втрати змісту.

Наше життя тісно повязане з музикою. Ми отримуємо задоволення від прослуховування музичних композицій, а також використовуємо її у відеороликах, кіно, презентаціях та рекламі. Тому здатність генерувати музику високої якості є значним кроком у житті людини. Проте досі немає відповіді на питання, чому одну послідовність нот люди сприймають як музику, а іншу ні. Це свідчить про те, що в музиці існують деякі інтуїтивно зрозумілі всім закономірності, які досі не вдалося математично строго сформулювати.

Існують два основні напрями розробки в сфері генерації музики. Перший вирішує задачу генерації музичних композицій без участі людини. Другий – спрямований на створення програмного забезпечення, яке допомагає людині створювати власні композиції. Розглядаючи детальніше тематику генеративного мистецтва, можна дійти висновку, що програмне забезпечення для автоматичної генерації музичних композицій є дуже перспективною розробкою.

Основною метою даної роботи є створення системи для автоматичної генерації музики, використовуючи рекурентні нейронні мережі.

1.Генеративне мистецтво

Автоматична генерація музики – це процес створення музичного твору з мінімальним втручанням людини.

Алгоритмічний підхід до створення музики існує вже кілька століть, починаючи з Гвідо д'Ареццо, який 1024 року

© О.С. Комарський, А.Ю. Дорошенко, 2022

винайшов перший алгоритм для складання музики [1]. Не зважаючи на те, що в докомп'ютерну епоху існувало кілька підходів до алгоритмічного створення музики, найяскравіші результати були отримані лише з появою комп'ютерів. Саме через величезні можливості, які пропонує обчислювальна техніка, алгоритмічні музичні композиції почали розквітати від початку 1950-х років та до сьогодні [2].

2.Модель нейронної мережі

Для побудови якісної нейронної мережі необхідно обрати відповідну базову архітектуру нейронної мережі [3]. Було вирішено обрати архітектуру AWD-LSTM, оскільки ця модель часто використовується для побудови нейронних мереж, які працюють у напрямку мовних досліджень [4].

Хоча багато моделей для генерації музики використовують невеликй діапазон нот, у даній реалізації модель використовує діапазон із 62 нот (цей діапазон охоплює більшість класичної музики) та дозволяє відтворювати будь-яку кількість нот та інструментів одночасно. Для навчання було використано великий датасет MIDI файлів із класичною музикою. Але, оскільки в моделі немає ніяких специфічних налаштувань для класичної музики, то можна використовувати датасет будь-якого музичного жанру.

3.Методи нот та акордів

Одним із можливих рішень для генерації музики – щоразу запитувати модель, чи грати їй зараз цю ноту для кожної з 88 клавіш фортепіано на кожному музичному кроці. Однак, оскільки кожна окрема нота здебільшого мовчить, то для нейронної мережі важко було б навчитись передбачати таку ситуацію для нот. Тому варто розглядати два можливі способи для рішення цієї проблеми.

Мовні нейронні мережі часто працюють на рівні символів, або на рівні слів. Так само для генерації музики, тут досліджується два аналогічних рівня – нот та акордів.

Акордом вважається кожна комбінація нот, які будь-коли зустрічалися в музичній композиції. Тобто сприймати акорд можна так само, як слова. Це означає, що можливо запитати нейронну мережу, яким буде наступний акорд (слово), надавши їй попередньої послідовності акордів (речення). Потрібно розуміти, що мається на увазі акорд, як будь-яка комбінація нот, що відтворюються одночасно. Не обов'язково це мають бути традиційні музичні акорди.

Для методу, який використовує ноти, вважаємо, що початок та кінець кожної ноти є різними словами. Наприклад «p28» та «endp28» у текстовому форматі MIDI є різними словами, та будуть використовуватись для початку та зупинки ноти №28.

В обох методах було використано дещо зменшений діапазон нот – 62 ноти замість повних 88 на фортепіано. Ноти які виходили з цього діапазону, було переміщено вниз або вгору на октаву.

В результаті експериментів були визначені характеристики кожного з двох методів.

Характеристика методу акордів:

1. Нейронна мережа добре генерує довгі музичні патерни, але не може вийти за межі навчальних даних та згенерувати щось нове.

2. Зазвичай для прийняття рішення щодо наступного акорду, найефективніше обирати найбільш імовірний прогноз на кожному часовому відрізку.

3. В результатах зустрічається дивний ефект, коли час від часу музична композиція перестрибує на іншу неочікувану ноту, але потім одразу продо-вжує грати мелодію в звичайному ритмі.

Характеристика методу нот:

1. Нейронна мережа створює гармонійні ритмічні малюнки, які можуть легко переходити один в другий.

2. Легко обробляє параметри тривалості натискання ноти, через що мелодія здається більш інтуїтивно зрозумілою для людини.

3. Зазвичай для прийняття рішення щодо наступної ноти найефективніше чергувати між собою три найбільш імовірні прогнози на кожному часовому відрізку.

Отже, в результаті цих порівнянь було прийнято рішення про використання методу нот як основного для генерації музики в системі.

4.Навчання нейронної мережі

Навчання моделі нейронної мережі потребує багато обчислювальних ресурсів та велику кількість якісних навчальних даних. Але, чим більше даних буде використовуватися, тим більше часу та обчислювальної потужності необхідно моделі для завершення навчання [5].

Для прискорення навчання були використані хмарні обчислення за допомогою ресурсів графічного процесора, оскільки такий підхід забезпечує більшу швидкість опрацювання даних, ніж центральний процесор. Це видно з результатів порівняння, зображених на рис. 1. Для порівняння були використані центральний процесор i7 8550 U на 4 ядра, та графічний процесор Geforce MX150 з 2 гігабайтами оперативної пам'яті. Як видно з діаграми, із зростанням кількості даних зростала швидкість їх обробки, особливо використовуючи графічний процесор, який за максимальної кількості даних був швидшим, аніж центральний процесор у 5,5 разів.

Вибір даних для тренування є важливою складовою будь-якої нейронної мережі, оскільки вона отримує всі свої «знання» про музику лише з навчальних даних. Тому, чим більше буде даних для навчання, та чим якісніше вони будуть, тим кращими будуть результати, згенеровані нейронною мережею. Також необхідно мати дані для тестування нейронної мережі, які повинні відрізнятися від даних для тренування, щоб визначати, чи змогла модель абстрагуватися від даних тренування та скласти нові мелодії, а не просто відтворювати дані.

До тренувальних даних є дві вимоги. Перша – музичні композиції мають бути в одному жанрі та написані людьми, а не іншою нейронною мережею. Друга вимога – мелодії мають приємно звучати. Оскільки зазвичай різні жанри музики для людей звучать по-різному, адже залежать від їхнього смаку, то було вирішено обрати для навчання класичну музику як найбільш нейтральний жанр. До того ж класична музика – це найдоступніший музичний жанр у форматі MIDI.

Для збору навчального датасету було написано декілька скриптів, за допомогою яких були зібрані файли з найбільшого сайту агрегатора класичної музики – «Classical Piano Midi Page», де представлені композиції таких композиторів як Моцарт, Бах, Бетховен та багато інших.

Для навчання моделі необхідно було перетворити зібрані MIDI-файли у формат, який зможе зрозуміти нейронна мережа. Файли MIDI надають інформацію про час початку гри та зупинки на кожну ноту, а також інформацію про музичний інструмент, значення гучності та темпу гри [6]. Але, оскільки цей формат має специфічні позначення, які не дуже зручно прогнозувати, було написано функцію, котраможе перекодувати їх до зручнішого текстового формату, який може зрозуміти модель нейронної мережі. Також на виході нейронної мережі

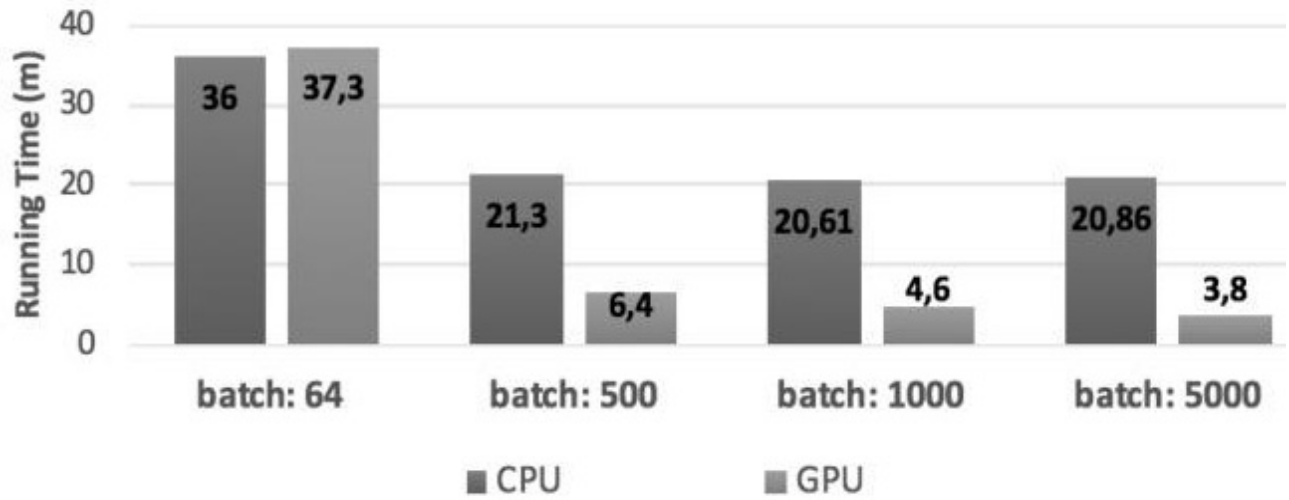


Рисунок 1. Порівняння швидкості навчання нейронної мережі з використанням CPU та GPU

музичні композиції подаються спочатку у вигляді власного текстового формату, а далі кодуються в формати MIDI та MP3 (рис. 2).

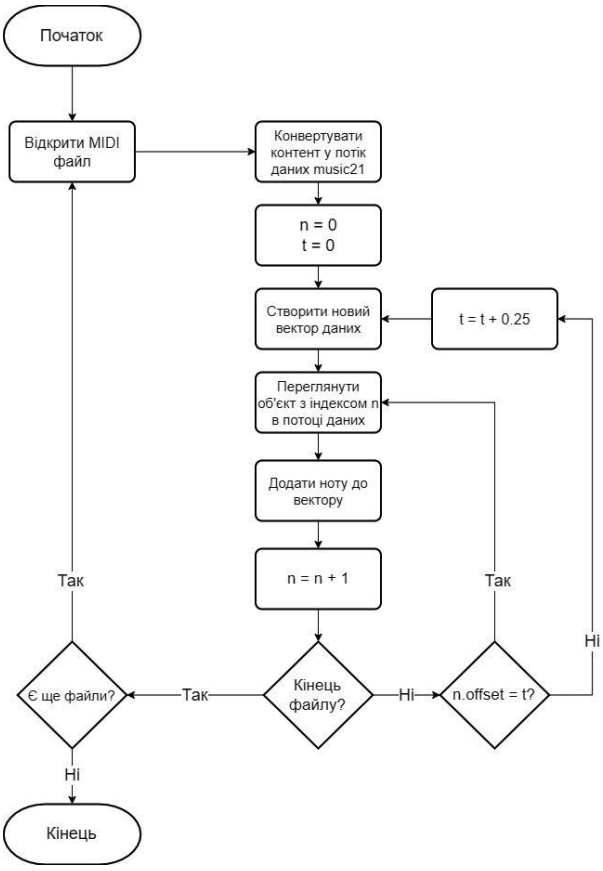


Рисунок 2. Блок-схема алгоритму перетворення формату MIDI у текст.

5.Результати

Під час тестування роботи нейронної мережі було досліджено вплив зміни деяких параметрів нейронної мережі на функцію втрати [4].

Експериментальним шляхом було визначено, що оптимальними характеристиками рекурентної нейронної мережі для генерації музики є мережа, яка складається з 4 LSTM шарів [4], кожен розмірністю по 600 нейронів зі значенням показника виключення (dropout) = 1 та кількістю епох 150.

Фінальний графік функції втрати нейронної мережі з обраними характеристиками зображено на рис. 3.

6.Оцінка якості нейронної мережі

Через те, що основною задачею є генерація музичних творів, то її якість не можна оцінити за об'єктивними характеристиками, оскільки оцінка краси, або естетичної цінності у творах мистецтва зводиться до індивідуальної суб'єктивної думки людей.

Для цього було розроблено спеціальне опитування, метою якого є оцінка наступних цілей, поставлених перед результатами генерації:

– музичні композиції повинні приємно та гармонійно звучати для людського вуха;

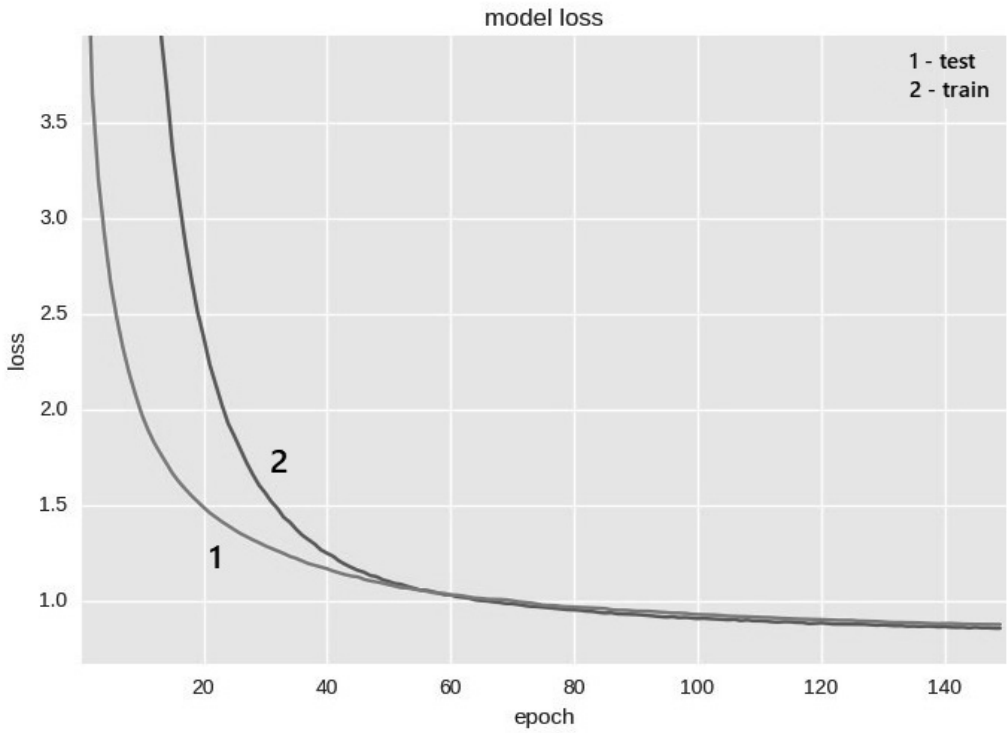


Рисунок 3. Графік функції втрати для фінальної моделі нейронної мережі.

– при прослуховуванні не має бути зрозумілим, чи мелодію було згенеровано нейронною мережею, чи людиною.

Для суб'єктивної оцінки якості музики згенерованої нейронною мережею, було підготовлено опитування, створене на платформі Google Forms. Тест було розроблено так, щоб можна було чітко зробити висновки та відповісти на два основних запитання про якість музичних композицій.

На початку проходження тесту для більш чіткої обробки результатів визначається група, до якої належить людина (фокус-група), яка буде проходити тест. Для відповіді було запропоновано декілька варіантів основних груп, що мають зміст у рамках тесту.

Наступним етапом є безпосереднє опитування щодо якості музичних композицій. Для цього спочатку надається можливість прослухати композицію, після чого пропонується відповісти на два запитання. Перше запитання стосується якості звуку, тобто суб'єктивної оцінки приємності звучання за шкалою від одного до десяти, де 1 – погано, 10 – дуже добре. Друге запитання стосується визначення автора твору – комп'ютер або людина. Усього 6 подібних етапів для 6 музичних творів.

Кожна музична композиція, яка наведена в опитуванні, – це, або згенерована музика нейронною мережею, що створена в рамках цієї роботи, або одна з композицій написаних людиною, яка була взята з навчального набору даних.

Також на початку опитування учасникам повідомляється, що принаймні одна з композицій була створена комп'ютером, а також принаймні одна є справжнім музичним твором, написаним одним із відомих композиторів.

Отже, опитування є також своєрідним «музичним тестом Тюринга», оскільки фокус-група одночасно оцінює якість творів, а також визначає, створений він людиною чи комп'ютером.

Опитування проводилось протягом тижня, за цей час у ньому взяли участь 55 людей. Опитування відбувалося в місцях, де потенційно можуть знаходитися люди з цільових груп, тобто в різних музичних спільнотах.

Переважно це були люди, які полюбляють слухати музику (43,6%), але не мають досвіду гри на музичних інструментах. На другому місці за кількістю людей у фокус-групі були ті, хто мають певний досвід гри на інструментах (32,7%). Решт - це учасники з музичною освітою (16,4%), або професійні музиканти, які постійно мають справу з музикою у роботі (7,3%). Детальний розподіл учасників опитування за музичним досвідом можна побачити на рис. 4.

Також були підраховані середні значення оцінки якості музичних композицій

Рисунок 4. Розподіл учасників опитування за їх музичним досвідом.

за результатами опитування, які наведені на рис. 5.

Отже, як бачимо з результатів опитування, загальна середня оцінка за трьома композиціями майже однакова, але дещо більша у творів написаних людиною. Проте, оскільки середні результати по кожній композиції майже на одному рівні, це означає, що мелодії, згенеровані нейронною мережею є якісними, оскільки подобаються фокус-групі так само, як і композиції, написані людиною. Також жодна з композицій не отримала оцінки нижче 3.

Наступним етапом обробки результатів опитування є підрахунок відповідей щодо визначення авторства композиції – людина чи комп'ютер. На рис. 6 бачимо зведену діаграму, яка показує у відсотках випадки, коли учасники опитування вірно або невірно визначали автора.

Отож, учасникам опитування було важко правильно визначити, хто є насправді автором прослуханих музичних творів – людина чи нейронна мережа. Згідно з діаграмою, на якій зведені результати опитування на це запитання, фокус-групі вдалося правильно відповісти лише в 58,2 % відповідей, що трохи краще, ніж звичайний шанс випадковості на запитання з двома варіантами відповіді (50%). Також ціка-

Група	Композиції згенеровані нейронної мережею			Композиції створені людиною		
Назва	Перша	Друга	Третя	Перша	Друга	Третя
Середня оцінка якості	5,54	6,62	6,12	6,32	7,12	5,5
Загальна оцінка якості за групою	6,09			6,31		

Рисунок 5. Середні показники якості музичних композицій за результатами опитування.

Рисунок 6. Результати опитування на питання про визначення автора музичного твору.

во, що 27,3 % опитуваних хибно вважали, музичний твір, створеним людиною, тоді як насправді він був згенерований нашою системою. Це свідчить про те, що учасникам було важко відрізнити згенеровані твори від написаних композиторами.

Висновки

У роботі була розглянута й реалізована можливість використання рекурентних нейронних мереж для задачі генеративного мистецтва. Були запропоновані та розглянуті два підходи, які використовують метод нот та метод акордів. У результаті досліджень основним був обраний метод нот, оскільки результатом його роботи були цікавіші та гармонійніші музичні композиції, на відміну від результатів, які вдалось отримати методом акордів. Також було здійснено порівняння швидкості навчання нейронної мережі з використанням центрального та графічного процесорів, за результатами якого графічний процесор обробляв дані швидше за центральний у 5,5 разів.

Для оцінки якості генерації музичних творів було проведено опитування фокус-групи, яке показало, що музика згенерована нейронною мережею, отримала майже такі ж оцінки, як і музика, написана людиною, що є відмінним результатом. Також було встановлено, що учасникам опитування було важко правильно визначати автора музичного твору, адже вони вірно визначали авторів лише в 58 % випадків.

References

1. *Nierhaus G.* (2009) Algorithmic Composition - Paradigms of Music Generation. Springer, Vienna. pp. 7-66.
2. *Wasserman P.* (1992) Neural Computing : Theory and Practice. pp. 180-185.
3. *Callan R.* (2001) The Essence of Neural Networks. pp. 50-65.
4. *Nielsen A.* (2015) Neural Networks and Deep Learning. p. 111.
5. *Kandel E., Schwartz J., Jessell T.* (2000) Principles of Neural Science 4th Edition. pp. 283-302.
6. Introduction to MIDI and Computer Music. URL:https://cecm.indiana.edu/361/midi.html

Отримано: 17.02.2022

***Про авторів*:**

Комарський Олександр Сергійович,
магістрант Національного
технічного університету України
«КПІ імені Ігоря Сікорського».
https://orcid.org/0000-0002-1005-6863

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор кафедри інформаційних систем
та технологій Національного технічного
університету України «КПІ імені Ігоря
Сікорського» та завідувач відділу теорії
комп'ютерних обчислень Інституту
програмних систем НАН України.
Кількість наукових публікацій
в українських виданнях – понад 190.
Кількість наукових публікацій
в зарубіжних виданнях – понад 80.
Індекс Хірша – 6.
http://orcid.org/0000-0002-8435-1451

Місце роботи авторів:

Національний технічний університет
України «Київський політехнічний
інститут імені Ігоря Сікорського»,
проспект Перемоги 37 та
Інститут програмних систем
НАН України, 03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559
E-mail: komarskiy33@gmail.com,
doroshenkoanatoliy2@gmail.com

ПРОБЛЕМИ ПРОГРАМУВАННЯ.
2022 – № 1

УДК 004.94; 004.4; 004.62

Спеціалізоване програмне забезпечення для моделювання динамічної консолідації віртуальних машин / Е.В. Жаріков, С.Ф.Теленик.

Для багатьох провайдерів хмарних послуг віртуальні машини залишаються базовою технологією віртуалізації обчислень. Віртуальні машини використовуються як для розміщення прикладних програмних засобів, так і для реалізації контейнерної віртуалізації. В умовах широкого використання віртуальних машин виникає необхідність розроблення спеціалізованого програмного забезпечення, що дозволяє визначати вплив параметрів моделей і методів управління на показники якості процесу консолідації, що дозволить запобігти виконанню експериментальних досліджень у виробничих умовах з метою оцінки нових стратегій управління ресурсами хмарного центру оброблення даних (ЦОД). Останніми роками в літературі запропоновано різні набори програмних інструментів і фреймворків для моделювання роботи ЦОД, забезпечуючи платформу та необхідні будівельні блоки для оптимізації процесу консолідації віртуальних машин. Моделі та програмні засоби моделювання процесів управління ресурсами ЦОД зазвичай не є вичерпними і вирішують конкретну проблему або завдання управління. Запропоноване у статті спеціалізоване програмне забезпечення моделювання дозволяє дослідити різні режими управління динамічною консолідацією віртуальних машин, забезпечує протоколювання результуючої інформації, такої, як показники продуктивності та діаграми навантажень, а також дозволяє визначати оптимальні параметри моделі для різних режимів роботи ЦОД, мінімізуючи кількість активних фізичних серверів та зменшуючи кількість порушень SLA.

Ключові слова: консолідація віртуальних машин, віртуалізація, хмарні обчислення, діаграма класів, діаграма послідовності.

UDC 004.94; 004.4; 004.62

Specialized software for simulating dynamic virtual machine consolidation / E. V. Zharikov, S. F. Telenyk.

For many cloud service providers, virtual machines remain the basic technology for computing virtualization. Virtual machines are used both to host application software and to implement container virtualization. Widespread use of virtual machines develops specialized software to determine the impact of model parameters on the quality of the consolidation process, which will prevent experimental research in production to evaluate new cloud data center resource management strategies. In recent years, there were many approaches in literature that offers various sets of software tools and frameworks for modeling data center processes, providing a platform and the necessary building blocks to optimize the process of consolidation of virtual machines. Models and software tools for modeling data center resource management processes are usually not exhaustive and solve a specific problem or management task. The specialized simulation software presented in the paper allows to investigate different control modes of virtual machines dynamic consolidation, provides a wide range of logging and debugging information using text and MS Excel files, such as performance indicators and workload diagrams, and allows to determine the optimal model parameters for various modes of data center operation, minimizing the number of active physical servers and reducing the number of SLA violations.

Keywords: virtual machine consolidation, virtualization, cloud computing, class diagram, sequence diagram

УДК 004.4′24

Автоматизована система управління запасами на ОС Android з використанням штрих кодів та QR-кодів. / Я.О. Гайдукевич, А.Ю. Дорошенко.

Розроблено програмний засіб для автоматизації запасами на основі ОС Android з використанням бази даних Firebase на мові програмування Java. У додатку використовується система зчитування штрих кодів та QR-кодів, яка забезпечує додавання товару на склад з усіма подробицями про обраний товар, що цікавить та перегляд всіх запасів на складі з їхньою ціною, категорією, ім'ям та кодом товару. Реалізовано авторизацію користувачів системи за допомогою відкритого стандарту FirebaseAuth. Створення єдиного сервісу для авторизації та реєстрації дозволило зробити систему масштабованою. У системі реалізовано пошук товару за кодом, а також перегляд запасів на складі з автоматичним підрахунком ціни. Основний сервіс бази даних - хмарна СУБД класу NoSQL, що дозволяє зберігати та синхронізувати дані між кількома клієнтами, передбачено API для шифрування даних. Крім цього, база даних залишається доступною тільки у додатку для уникнення несанкціонованого доступу та редагування. В ході розробки було прийнято рішення зробити автоматизовану систему управління на девайсі ОС Android, тому що промислова діджиталізація – саме через масове впровадження розумних пристроїв (smart devices) - вимагає мобільності та прискореного темпу роботи із запасами у різних сферах товарообігу. Проведено тестування розробленого програмного засобу.

Ключові слова: ОС Android, Java, Firebase, авторизація, архітектура API, маштабовані системи автоматизації, NoSQL , СУБД , API для шифрування даних.

UDC 004.4′24

Automated inventory management system on Android using barcodes and QR-codes / Y.O. Haidukevych, A.Yu. Doroshenko.

A software is developed for inventory automation based on Android using the Firebase database in the Java programming language. The appendix uses a system of reading barcodes and QR-codes, which provide the addition of goods to the warehouse with all the details of the selected product and information on all stocks in the warehouse with their price, category, name and code goods. Implemented authorization of system users using the open FirebaseAuth standard. Creating a single service for authorization and registration allows to make the system scalable. The system implements the search for goods by code, as well as viewing stocks in the warehouse with automatic price calculation. The main service of the database is a cloud database of the NoSQL class, which allows storing and synchronizing data between several clients, there is an API for data encryption. In addition, the database remains available only in the application to avoid unauthorized access and editing. While development, it was decided to make an automated control system on the Android OS device, because industrial digitalization (precisely because of the mass introduction of smart devices) requires mobility and accelerated processing of stocks at various stages of turnover. Testing of the developed software is carried out.

Keywords: Android, Java, Firebase, authorization, API architecture, scalable automation systems, NoSQL, DBMS, API for data encryption.

УДК 681.3

Використання систем організації знань на основі онтологій у WIKI – ресурсах / Ю.В.Рогушина.

У роботі розглядаються теоретичні основи систем організації знань (СОЗ) в інтелектуальних застосуваннях на основі онтологій. Метою даного дослідження є аналіз застосування різних типів СОЗ для організації та вдосконалення бази знань семантизованих Wiki-ресурсів, які містять гетерогенний мультимедійний контент великого обсягу та мають складну структуру, що інтегрує знання із різних ПрО. Розглянуто діалекти мови подання онтологій OWL та їхню виразність для подання окремих випадків онтологій, що використовуються у СОЗ. Проаналізовано критерії класифікації СОЗ та сфери їх застосування. Запропоновано формальну модель онтології семантизованого Wiki-ресурсу та засоби реалізації різних видів відношень між об'єктами у середовищі Semantic MediaWiki з використанням шаблонів. Розглядаються проблеми доступу до інформації у цих ресурсах з точки зору СОЗ та наводяться методи й засоби вирішення цих проблем. Запропоновано реалізацію згадуваного підходу на прикладі портальної версії Великої Української Енциклопедії (е-ВУЕ).

Ключові слова: система організації знань, онтологія, тезаурус, Wiki-ресурс.

UDC 681.3

Use of Ontology-based knowledge Organization Sysytems for WIKI Resources / J. Rogushina.

The paper considers the theoretical foundations of knowledge organization systems (KOSs) in intelligent ontology-based applications. The aim of this study is to analyze the use of different types of KOSs to organize and improve the knowledge base of semantic Wiki resources that contain heterogeneous multimedia content of large volume and have a complex structure integrated knowledge from different domains. The dialects of the OWL ontology representation language and their expressiveness for representing special cases of ontologies used in KOSs are considered. The criteria for the classification of KOSs and sphere of their usage are analyzed. Formal model of ontology for semantic Wiki resource is proposed. This model is integrated with various implementing means for different types of relations between objects in the Semantic MediaWiki environment based on templates. Problems of access and retrieval of information in these resources and methods of their solving from the KOSs point of view are considered. The software implementation of the proposed approach with the example of the portal version of the Great Ukrainian Encyclopedia (e-VUE) is realized.
The urgency of the problem intensifies by the need for national information resources in martial law situation, for which the determining factors of effective information processing are both the ability to obtain satisfaction of complex information needs and the relevance of the information obtained. This increases the importance of official government portals that integrate reliable data from various fields of knowledge and prevent possible misrepresentation (both accidental and malicious) of information in resources with open content generation .

Keywords: knowledge organization system, ontology, thesaurus, Wiki-resource.

УДК 004.94

60 років базам даних/ В.А.Резніченко.

Наводиться огляд досліджень і розробок баз даних з моменту їх виникнення в 60-х роках минулого століття і по сьогодні. Виділяються наступні етапи: виникнення і становлення, бурхливий розвиток, епоха реляційних баз даних, розширені реляційні бази даних, пост-реляційні 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 наводяться результати наукової діяльності Е. Кодда, теорія залежностей і нормальних форм, мови запитів, експериментальні дослідження і розробки, оптимізація та стандартизація, управління транзакціями. Етап розширених реляційних баз даних присвячений опису темпоральних, просторових, дедуктивних, активних, об'єктних, розподілених та статистичних баз даних, баз даних масивів, а також машин баз даних і сховищ даних. На наступному етапі розкрита проблематика постреляційних баз даних, а саме, NOSQL-, NewSQL- і онтологічних баз даних. Шостий етап присвячено розкриттю причин виникнення, характерних властивостей, класифікації, принципів роботи, методів і технологій великих даних. Нарешті в останньому розділі подається короткий огляд досліджень і розробок по базах даних в Радянському Союзі.

Ключові слова: Типи баз даних: ієрархічна, мережева, реляційна, навігаційна, темпоральна.

UDC 004.94

60 Years of Databases/ V.A. Reznichenko.

The article provides an overview of research and development of databases since their appearance in the 60s of the last century to the present time. The following stages are distinguished: the emergence formation and rapid development, the era of relational databases, extended relational databases, post-relational databases and big data. At the stage of formation, the systems IDS, IMS, Total and Adabas are described. At the stage of rapid development, issues of ANSI/X3/SPARC database architecture, CODASYL proposals, concepts and languages of conceptual modeling are highlighted. At the stage of the era of relational databases, the results of E. Codd's scientific activities, the theory of dependencies and normal forms, query languages, experimental research and development, optimization and standardization, and transaction management are revealed. The extended relational databases phase is devoted to describing temporal, spatial, deductive, active, object, distributed and statistical databases, array databases, and database machines and data warehouses. At the next stage, the problems of post-relational databases are disclosed, namely, NOSQL-, NewSQL- and ontological databases. The sixth stage is devoted to the disclosure of the causes of occurrence, characteristic properties, classification, principles of work, methods and technologies of big data. Finally, the last section provides a brief overview of database research and development in the Soviet Union.

Keywords: Database types: hierarchical, network, relational, navigational, temporal, spatial. spatio-temporal, spatio-network, moving objects, deductive, active, object-oriented, object-relational, distributed, parallel, arrays, statistical, multidimensional, database machines, data warehouse, NoSQL, key-value , triple store, column-oriented, document- oriented, graph-oriented, multimodal, cloud, scientific, multi-valued, XML, NewSQL, ontological, Big Data.

УДК 004.042

Аналітичне сховище для великих потокових даних / Тюрін В.О. Дорошенко А.Ю., Савчук О.В.

Розроблено концепцію організації аналітичного сховища даних, що включає метод взаємодії продюсерів даних зі сховищем, методи контролю схеми даних та потокової передачі даних, метод зберігання початкових даних, методи обробки даних та надання безпечного доступу до даних. Розглянуто існуючі на ринку стандарти, а саме: SDLF – провідний стандарт, рекомендований AWS, IronSource DL з використанням Upsolver, SimilarWeb DL з використанням Upsolver. Проведено порівняльний аналіз (здебільшого з SDLF, бо його реалізація є відкритою, а деталі реалізацій приватних компаній приховуються).
Детально оглянуто переваги запропонованої концепції над існуючими стандартами. Наведено рекомендації щодо способу інтеграції концепції із застосунками з контролю схеми даних. Розроблено сервіс потокової передачі даних за допомогою Apache Beam мовою Java. Спроєктовано та розроблено архітектуру сховища для аналітики, модель управління схемою даних та модель безпечного надання доступу до даних.
Проведені розробки та дослідження можна покращувати шляхом описання досвіду впровадження концепції у бізнесах, а також за рахунок збору та систематизації знань про стандарти, що будуть створені.

Ключові слова: безсерверні аналітичні сховища даних, BigQuery, AWS, GCP, ETL, повідомлення, потокова передача даних.

UDC 004.042

Analytical store for streaming data with huge volume / V.O. Tiurin, A.Yu. Doroshenko, O.V. Savchuk.

A concept for organizing an analytical data warehouse has been developed, which includes a method of interaction between data producers and a repository, a method of data circuit control, a method of data streaming, a method of storing initial data, a method of data processing and a method of providing secure data access. Other concepts on the market are discussed, namely: SDLF as the leading standard recommended by AWS, IronSource DL using Upsolver, SimilarWeb DL using Upsolver. A comparative analysis was conducted (mostly with SDLF, as its implementation is open, and the implementation by private companies is hidden). The advantages of the proposed concept over the existing ones are examined in detail. Recommendations on how to integrate the concept with data schema control applications are given. A service for streaming data using Apache Beam in Java has been developed. A repository architecture for analytics was designed and developed. A data schema management model was developed as well as a data schema management model and a model for secure access to data. The research that has been conducted can be improved by the experience of implementing the concept in business, as well as by collecting and systematizing knowledge about other standards that will be created.

Keywords: serverless, analytical datastore, BigQuery, AWS, GCP, ETL, message, data streaming.

УДК 517.9:621.325.5:621.382.049.77

Проблеми розробки архітектури сучасних когнітивних радіолокаційних систем / М. Косовець, Л. Товстенко.

Розглянуто проблему розробки архітектури сучасних когнітивних радіолокаційних систем із використанням технологій штучного інтелекту. Основною відмінністю від традиційних систем є використання навченої

UDC 517.9:621.325.5:621.382.049.77

The problems of developing the architecture of modern cognitive radar systems / M. Kosovets, L. Tovstenko.

The problem of developing the architecture of modern cognitive radar systems using artificial intelligence technologies is considered. The main difference from traditional systems is the use of a trained neural network. The heterogeneous

нейронної мережі. Гетерогенна багатопроцесорна система перебудовується в процесі розв'язування задачі, забезпечуючи надійність і вирішення різних типів задач одного класу і глибоке навчання нейронної мережі в режимі реального часу. Така архітектура сприяє впровадженню когнітивних технологій, які враховують вимоги по призначенню, вплив зовнішніх і внутрішніх факторів. Глибоке навчання нейронної когнітивної мережі радарних датчиків є функцією штучного інтелекту, який моделює роботу людського мозку таким чином, що обробляє дані та створює шаблони, які використовуються у прийнятті рішень. Система виявлення вчиться виявляти зміни не тільки в рівнях сигналу, а й у формі та параметрах сигналу. Експерименти показали, що система виявлення змін радіолокаційної інформації на основі нейронної мережі з глибоким навчанням є оптимальною для розробки сенсорних мережевих додатків і може бути успішно реалізована на доступних технологічних платформах.

Ключові слова: штучний інтелект, нейронна когнітивна мережа, сенсорні мережеві додатки.

multiprocessor system is rebuilt in the process of solving the problem, providing reliability and solving various types of problems of one class and deep learning of the neural network in real time. This architecture promotes the introduction of cognitive technologies that take into account the requirements for the purpose, the influence of external and internal factors. Deep learning of the neural cognitive network of radar sensors is a function of artificial intelligence that simulates the work of the human brain in such a way that it processes data and creates templates which use in decision making. The detection system learns to detect changes not only in signal levels, but also in the shape and parameters of the signal. Experiments have shown that the system of detecting changes in radar information based on neural network with deep learning is optimal for the development of sensor network applications and can be successfully implemented on available technology platforms.

Keywords: Perception-Action Cycle, Artificial Intelligence, Signal to Noise Ratio, Active Electronically Scanned Array, Environmental Dynamic Database, Signal to Noise Ratio, Radar Resource Management, multiprocessor.

УДК 004.89

Модель рекурентної нейронної мережі для генерації музики / О.С. Комарський, А.Ю. Дорошенко.

У роботі розглядається можливість генерації музичних композицій, використовуючи рекурентні нейронні мережі. Запропоновано та розглянуто два підходи генерації музичних творів, а саме за допомогою метода нот та метода акордів. Проведено дослідження обох методів та сформовані їх переваги та недоліки, в результаті чого було вирішено використовувати метод нот, як основний для генерації музики. Детально описано процес пошуку та обробки даних для навчання музичної нейронної мережі, детально розглянуто алгоритм перетворення даних з формату MIDI у власний текстовий для використання в нейронній мережі. Також описано процес навчання нейронної мережі та порівняно швидкість навчання, використовуючи графічний та центральний процесори. Було визначено, що навчання відбувається швидше із використанням графічного процесору

UDC 004.89

Recurrent neural network model for music generation / O.S. Komarskiy, A.Yu. Doroshenko.

The paper considers the possibility of generating musical compositions using recurrent neural networks. Two approaches to the generation of musical works are proposed and considered, namely using the method of notes and the method of chords. The research of both methods was carried out, and their advantages and disadvantages were formulated. As a result it was decided to use the method of notes as the main one for music generation. The process of searching and processing data for learning a music neural network is described in detail, the algorithm for converting data from MIDI format to your own text for use in a neural network is considered in detail. The learning process of the neural network was also described, and the learning speed was compared using GPUs and CPUs, as a result of which it was determined that learning takes place faster using a graphics processor,

в деяких випадках у 5,5 разів. У результаті тестування роботи нейронної мережі було встановлено, що оптимальними характеристиками рекурентної нейронної мережі для генерації музики – є мережа, що складається з 4 LSTM шарів, кожен розмірністю у 600 нейронів. Оскільки генерацію музики не можна оцінити за об'єктивними характеристиками, то для оцінки якості було проведено спеціальне опитування серед фокус-групи, яке показало, що музика, згенерована нейронною мережею, отримала майже такі ж оцінки, як і музика, написана людиною, що можна вважати відмінним результатом. Також було з'ясовано, що учасникам опитування було важко правильно визначати автора музичного твору оскільки вони правильно визначали авторів лише в 58 % випадків. Запропоноване рішення дозволяє легко генерувати музичні твори без втручання людини.

Ключові слова: генерація музики, машинне навчання, рекурентні нейронні мережі, LSTM, RNN.

in some cases 5.5 times. As a result of testing the operation of the neural network, it was determined that the optimal characteristics of the recurrent neural network for music generation is a network consisting of 4 LSTM layers, each with a dimension of 600 neurons. As music generation cannot be assessed by objective characteristics, a special focus group survey was conducted to assess quality. It shows that music generated by a neural network received almost the same marks as music. written by a man. It should be considered as a great result. It was also determined that it was difficult for the survey participants to correctly identify the author of a musical work, since they correctly identified the authors in only 58% of cases. The proposed solution allows to easily generate musical compositions without human intervention.

Keywords: music generation, machine learning, recurrent neural networks, LSTM, RNN.

ДО УВАГИ АВТОРІВ!

У журналі «Проблеми програмування» публікуються наукові матеріали, які раніше не публікувалися в інших виданнях.

Мова статті: українська, англійська.* Обсяг статті - від 6 до16 сторінок формату А4.

Документ зберігається у форматі doc або docx. Ім'я подається транслітерацією, як прізвище автора (авторів), наприклад, "Petrenko.doc".

Автори можуть користуватися електронною поштою і також телефаксом для ділової переписки та передачі до редакції тексту статті та правки при коректурі. E-mail редакції: alengoro@isofts.kiev.ua. FAX: +380 (44) 526 6263, Телефон: 526 5065.

1. Оформлення файлу з текстом статті.

При підготовці файлу використовуються: стиль нормальний (звичайний) або normal; шрифт Times New Roman, розмір шрифта 12 пт.; міжрядковий інтервал – 1,0; абзацний відступ -1,25 см; вирівнювання – по ширині. У тексті не допускається вирівнювання пропусками; розстановка переносів – автоматична. Формат паперу А4, розміри полів документа – 20 мм. Текст статті після анотації має бути оформлений **у 2 колонки,** ширина яких – 7,86 см, а пробіл між ними – 1,27 см.

2. Послідовність розміщення та оформлення матеріалу статті.

УДК: індекс за універсальною десятковою класифікацією.

Автори: ініціали та прізвища авторів, курсив (світлий).

Заголовок 1 **(назва статті):** не містить абревіатур та строго відповідає змісту статті. Шрифт 15 пт, напівжирний, регістр верхній.

Анотація **(мовою статті):** 50-100 слів, не містить абревіатур, зрозумілих із змісту статті. Шрифт 10 пт, звичайний.

Ключові слова **(мовою статті):** не більше10 слів, не містить абревіатур, зрозумілих із змісту статті, подаються в називному відмінку, розділені комами. Шрифт 10 пт, звичайний.

Заголовок 2 **(назва розділу):** шрифт 14 пт, напівжирний; абзац із центральним вирівнюванням, без переносів. Заголовки нижчого рівня (пункти і т.п.) у самостійний абзац не виділяються і проходять першим реченням текстового абзацу, шрифт 12 пт, напівжирний.

Основний текст статті має такі необхідні елементи:

постановка проблеми в загальному вигляді і її зв'язок з важливими науковими або практичними завданнями;

аналіз останніх досліджень і публікацій, у яких розпочато рішення даної проблеми і на які спирається автор, виділення невирішених раніше частин загальної проблеми, яким присвячується дана стаття;

формулювання цілей статті (постановка задачі);

виклад основного матеріалу дослідження з повним обґрунтуванням отриманих наукових результатів;

висновки з даного дослідження і перспективи подальших розробок у даному напрямку;

подяка (за наявності такої).

Формули створюються в редакторі Microsoft Equation 3.0 або MathType. Формули, на які є посилання в тексті, повинні мати наскрізну нумерацію. Номер формули друкується в круглих дужках біля краю правого поля. Розмір основного шрифту редактора формул – 12 пт. Розміри символів у формулах: звичайний – 12 пт, великий індекс – 9 пт, дрібний індекс – 7 пт, великий символ – 18 пт, дрібний символ – 11пт. Не допускається масштабування формульних об´єктів.

Рисунки мають бути створені вбудованим редактором Word Picture або експортовані з прикладних програм Windows у графічних форматах (bmp, pcx, gif, jpg або tif). Рисунки розташовуються по центру. Нумерація рисунків здійснюється відповідно до порядку згадування у тексті. Нумеровані підписи розміщуються під рисунком з позначенням «Рис. », далі вказується номер рисунка і текст підпису.

Таблиці мають бути підготовлені стандартним вбудованим в Word інструментарієм "Таблиця". Таблиці нумеруються за порядком згадування. На номер таблиці повинно бути

посилання в тексті. Номер таблиці вказується в окремому рядку з вирівнюванням по правій стороні (наприклад, «Таблиця 1»). Назви таблиць розміщуються над таблицею з вирівнюванням по центру. Мінімальний розмір шрифту в таблицях – 11 пт.

Література*:** нумерований список джерел згідно ДСТУ 8302:2015 від 01.07.2016 р., шрифт 11 пт, відступ: спеціальний, навислий, 0,63 см. Джерела з заголовками на латиниці наводяться без перекладу. Інші джерела подаються мовою оригіналу. Приклади оформлення бібліографічних посилань згідно з вимогами ***Harvard Style наведені в багатьох публікаціях, наприклад: http://www.staffs.ac.uk/assets/harvard_referencing_examples_tcm44-39847.pdf

***Дані про авторів*:** мають починатися рядком "Про авторів:", напівжирний курсив. Далі вказуються для кожного з авторів ПІБ повністю, наукове звання, посада, адреса, кількість публікацій в українських виданнях (приблизна), кількість публікацій в зарубіжних індексованих виданнях (приблизна), індекс Хірша (за наявності), обов'язково номер ORCID (сайт ORCID http://orcid.org/).

***Дані про місце роботи авторів*:** починаються рядком "Місце роботи авторів:", напівжирний курсив. Далі вказуються місце роботи, адреса, телефон, факс, електронна пошта, контактний телефон.

3. Оформлення файлу з анотаціями.

Файл з анотаціями містить інформацію двома мовами (наприклад, якщо стаття написана українською мовою, то анотація та ключові слова – англійською і українською мовами) та має бути оформлений у дві колонки: УДК (шрифт – 8 пт); назва статті (шрифт – 12 пт, напівжирний); прізвища та ініціали авторів (шрифт – 12 пт); текст анотації, ключові слова (шрифт – 10 пт).

Вимоги до анотації англійською мовою: обсяг від 100 до 250 слів, інформативність, оригінальність (не є калькою української анотації), змістовність (відображає основний зміст статті і результати досліджень), структурованість (дотримується логіки опису результатів у статті).

Документ зберігається у форматі doc або docx. Ім´я подається транслітерацією, як прізвище автора (авторів), наприклад, "Petrenko_Annot.doc".

*16.07.2020 р. набули чинності положення Закону України «Про забезпечення функціонування української мови як державної». Відповідно до статті 22 «Державна мова у сфері науки» у наукових виданнях не повинно бути вміщено матеріалів іншими мовами, окрім державної, англійської та мов ЄС.

Примітка: Підписний індекс журналу «Проблеми програмування» – **90853**.