НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ISSN 1727-4907

ПРОБЛЕМИ ПРОГРАМУВАННЯ

НАУКОВИЙ ЖУРНАЛ

PROBLEMS OF PROGRAMMING

SCIENTIFIC JOURNAL

2021
№ 3

ТЕМИ ВИПУСКУ:

- ***Інструментальні засоби та середовища програмування***
- ***Інформаційні системи***
- ***Інформатизація наукових досліджень***
- ***Моделі та засоби систем баз даних і знань***

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ПРОБЛЕМИ ПРОГРАМУВАННЯ

науковий журнал

Головний редактор
Андон Пилип Іларіонович
академік НАН України,
директор Інституту програмних систем НАН України

Інститут програмних систем НАН України
проспект Академіка Глушкова, 40,
корп. 5 03187, Київ-187
Тел.+380 (44) 526 5507
E-mail: andon@isofts.kiev.ua
http://www.pp.isofts.kiev.ua

Редакційна колегія

Адреса для кореспонденції
Інститут програмних систем НАН України
Проспект Академіка Глушкова, 40
03187, Київ-187

Тел.: +380 (44) 526 5065
Факс: +380 (44) 526 6263
E-mail: iss@isofts.kiev.ua

Затверджено до друку вченою радою Інституту програмних систем НАН України.
Протокол № 7 від 16.09.2021 р.

Редактор З.В. Єгорова
Комп'ютерна верстка В.П. Бумажний

Підписано до друку 17.09.2021. Формат 60x84/8. Папір офс. Ум. друк. арк. ___
Обл.-вид. арк.___ Тираж 120 прим. Ціна договірна. Замовл.

Віддруковано ТОВ «Про формат»
вул. Маршала Жукова, 45 б, м. Київ, 02166

НАЦІОНАЛЬНА
АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ПРОБЛЕМИ ПРОГРАМУВАННЯ

науковий журнал

№ 3 ***липень - вересень*** ***2021***

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

NATIONAL ACADEMY
OF SCIENCES OF UKRAINE
INSTITUTE OF SOFTWARE SYSTEMS

PROBLEMS OF PROGRAMMING

scientific journal

№ 3 ***July – September*** ***2021***

Founded in March, 1999

CONTENTS

УДК 004.272.26 + 004.021 + 004.032.26 https://doi.org/10.15407/pp2021.03.003

І.З. Ашур, А.Ю. Дорошенко

РОЗПОДІЛЕНА РЕАЛІЗАЦІЯ МЕТОДУ НЕЙРОЕВОЛЮЦІЇ НАРОСТАЮЧОЇ ТОПОЛОГІЇ

У статті запропонована нова розподілена реалізація методу нейроеволюції наростаючої топології, яка, за наявності достатніх обчислювальних ресурсів, дозволяє радикально збільшити швидкість знаходження оптимальних конфігурацій нейронних мереж. З метою оптимізації продуктивності рішення, рівномірного розподілу завдань між вузлами та оптимального використання обчислювальних ресурсів була реалізована підтримка пакетної оцінки геномів. Експериментальна перевірка нової реалізації засвідчує, що, використовуючи запропоноване розподілене рішення, швидкість виконання методу нейроеволюції наростаючої топології в частині оцінювання згенерованих нейронних мереж на прикладі розглянутого завдання і середовища може зростати на декілька порядків.

Ключові слова: NEAT, нейроеволюція наростаючої топології, штучні нейронні мережі, навчання з підкріпленням, генетичні алгоритми, розподілені обчислення, хмарні обчислення.

Вступ

Попри сильні сторони методу нейроеволюції наростаючої топології, якот можливість його застосування в завданнях, де важко обрати функцію витрат і топологію нейронної мережі, однією з проблем нейроеволюції і, зокрема, методу нейроеволюції наростаючої топології, є повільна конвергенція до оптимальних результатів, особливо у випадку роботи з комплексними та складними середовищами. Ця робота пропонує розподілену реалізацію методу нейроеволюції наростаючої топології, що, за наявності достатніх обчислювальних ресурсів, дозволяє радикально збільшити швидкість знаходження оптимальних конфігурацій нейронних мереж.

Нейроеволюція

Нейроеволюція – форма машинного навчання, яка використовує еволюційні алгоритми для генерації штучних нейронних мереж, їхніх параметрів, топологій і правил. Головна перевага нейроеволюції полягає в можливості її ширшого застосування порівняно з навчанням з учителем, яке вимагає розмічених коректних пар вхідних та вихідних даних для тренування. На відміну від цього, нейроеволюція потребує лише можливості оцінити ефективність згенерованої мережі на будь-якому етапі навчання. Наприклад, результативність гіпотетичної ігрової партії у вигляді перемоги одного чи іншого гравця можна легко оцінити без надання розмічених даних про бажані або ефективні ігрові стратегії [1].

Нейроеволюцію часто розуміють як частину парадигми навчання з підкріпленням, що може бути протиставлена загальноприйнятим методам глибинного навчання, які використовують градієнтний спуск на штучних нейронних мережах із фіксованою топологією [2].

NEAT

Одне з головних питань нейроеволюції полягає у використанні переваг еволюції топології нейронної мережі поряд з еволюцією ваги з'єднань її вузлів. Нейроеволюція наростаючої топології (NeuroEvolution of Augmenting Topologies, NEAT) – генетичний алгоритм знаходження штучних нейронних мереж шляхом еволюції (нейроеволюційний метод), розроблений 2002 року дослідником Кеном Стенлі (Ken Stanley) [3].

Традиційно топологію нейронної мережі обирає людина-експериментатор, а значення ваги з'єднань між вузлами-нейронами отримують у процесі навчання. Це призводить до ситуації, коли необхідно застосувати підхід проб та помилок для того, аби знайти вдалу топологію для поставленого завдання. NEAT – приклад алгоритму, що шляхом еволюції одночас-

но змінює і топологію, і значення ваги з'єднань штучної нейронної мережі. Такі алгоритми належать до класу TWEANN (Topology and Weight Evolving Artificial Neural Network) [4].

Алгоритм нейроеволюції наростаючої топології починає процес еволюції зі штучної нейронної мережі, подібної до перцептрона, з лише вхідним та вихідним заздалегідь визначеними шарами вузлів-нейронів. Із плином дискретних кроків нейроеволюції складність топології нейронної мережі може зростати шляхом створення нових вузлів-нейронів у присутніх шляхах з'єднань або внаслідок створення нових з'єднань між попередньо роз'єднаними нейронами.

Метод нейроеволюції наростаючої топології перевершує найкращі методи нейроеволюції фіксованих топологій у завданнях навчання з підкріпленням. Нейроеволюція наростаючої топології демонструє високі показники ефективності завдяки використанню принципового методу кросинговеру різних топологій, захисту структурної інновації внаслідок видоутворення та інкрементального росту з мінімальних структур [5-7].

SharpNEAT

SharpNEAT – це .NET C# реалізація еволюційного алгоритму, а саме алгоритму еволюції нейронних мереж. Еволюційний алгоритм використовує еволюційні механізми мутації, генетичної рекомбінації та відбору для пошуку нейронних мереж, функціонал та поведінка котрих задовольнила би попередньо визначене завдання. Реалізація SharpNEAT розроблена дослідником Коліном Гріном (Colin Green) [8].

Алгоритм NEAT та реалізація SharpNEAT виконують пошук не тільки структури нейронної мережі, тобто її вузлів і з'єднань між ними, а й ваги цих з'єднань. Ця особливість вдало виділяє алгоритм нейроеволюції наростаючої топології з-поміж інших сучасних технік нейроеволюції та технік навчання з підкріпленням.

Фреймворк і демонстраційне програмне забезпечення SharpNEAT надає набір вбудованих задач-прикладів для демонстрації роботи алгоритму нейроеволюції. Реалізація SharpNEAT є модульною, що дозволяє легко вносити зміни, розширювати та повторно використовувати її частини. SharpNEAT створений, зокрема, для задоволення інтересів питань біологічної еволюції та границь можливостей нейроеволюції в зрізі рівня складності проблем, рішення котрих можуть запропонувати алгоритми нейроеволюції. На рис. 1 зображено високорівневу діаграму основних модулів оригінального проєкту SharpNEAT.

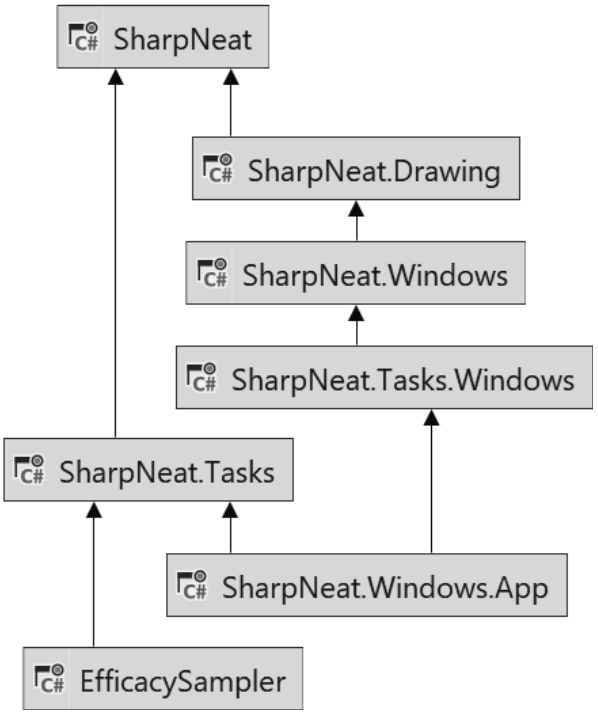


Рисунок 1. Високорівнева діаграма основних модулів оригінального проєкту SharpNEAT

- Модуль SharpNeat є базовим модулем із головними інтерфейсами й реалізацією функціоналу, пов'язаного зі штучними нейронними мережами та графами, функціями активації нейронів, оцінкою придатності геномів і феномів, алгоритмом нейроеволюції, експериментами, серіалізацією та десеріалізацією.

- Модуль SharpNeat.Tasks – модуль із вбудованими задачами навчання з підкріпленням.

- Модуль SharpNeat.Drawing відповідає за вбудований функціонал рендерингу штучних нейронних мереж.

- Модуль SharpNeat.Windows відповідає за високорівневі абстракції графічного інтерфейсу SharpNeat.

- Модуль SharpNeat.Windows.App відповідає за реалізацію графічного інтерфейсу SharpNeat для сімейства операційних систем Microsoft Windows.

- Модуль EfficacySampler відповідає за збір метрик ефективності алгоритму нейроеволюції наростаючої топології: часу виконання, кількості поколінь, показників придатності та складності тощо.

На рис. 2 зображено високорівневу блок-схему алгоритму нейроеволюції наростаючої топології в реалізації програмного фреймворку SharpNEAT.

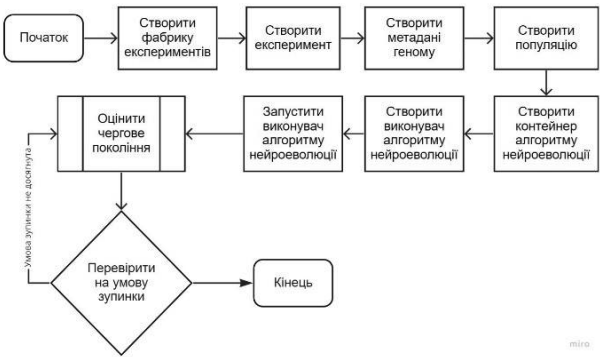


Рисунок 2. Високорівнева блок-схема алгоритму нейроеволюції наростаючої топології в реалізації програмного фреймворку SharpNEAT

Розподілені обчислення

Розподілені обчислення – це одна зі сфер комп'ютерних наук, що вивчає розподілені системи [9]. Розподілена система – це система, компоненти котрої знаходяться на різних комп'ютерах у мережі, комунікація і кооперація між якими відбувається шляхом обміну повідомленнями. Компоненти такої системи взаємодіють один з одним задля досягнення певної спільної мети.

До основних характеристик розподілених систем відносимо: одночасність роботи компонентів, відсутність глобальної синхронізації та незалежність відмови компонентів. Прикладами розподілених систем є системи з сервісно-орієнтованою архітектурою, масові багатокористувацькі онлайн-ігри, системи з одноранговою (peer-to-peer) архітектурою. Під розподіленими обчисленнями також розуміємо спосіб розв'язання трудомістких обчислювальних завдань із використанням багатьох комп'ютерів, об'єднаних у мережу.

Розподілені обчислення є окремим випадком паралельних обчислень, де одне обчислювальне завдання вирішується за допомогою декількох процесорів на одному або кількох комп'ютерах. Однією з вимог до завдання, яке вирішують за допомогою розподіленої системи, є можливість його розділення на підзавдання, що їх можуть обчислювати паралельно. Комп'ютерна програма, яку виконують на розподіленій системі, називається розподіленою. Існує багато реалізацій механізму комунікації між компонентами таких програм: HTTP, RPC тощо [9-12].

Розподілений алгоритм

Розподілений алгоритм – це алгоритм, створений для виконання на апаратному забезпеченні із взаємопов'язаними центральними процесорами. Розподілені алгоритми використовуються в різноманітних галузях розподіленого обчислення: телекомунікації, обчислювальних науках, розподіленій обробці даних, автоматизації виробництва.

Розподілені алгоритми – один із типів паралельних алгоритмів. Як правило, такі алгоритми виконуються одночасно, окремі частини алгоритму паралельно виконуються на незалежному обчислювальному апаратному та програмному забезпеченні з обмеженим знанням про діяльність інших частин алгоритму. Одна з найбільших проблем проєктування та реалізації розподілених алгоритмів полягає у втіленні вдалої координації поведінки незалежних частин алгоритму, включно з обробкою відмов незалежних вузлів і каналів їхньої комунікації.

Дизайн рішення

Наша робота подає високорівневий огляд розподіленої реалізації методу нейроеволюції наростаючої топології на основі реалізації SharpNEAT.

На рис. 3 зображено обчислювальні вузли під час роботи програми, а також

компоненти та об'єкти, які можуть бути виконані на цих вузлах. Для візуалізації використовуємо UML діаграму розгортання. Діаграма розгортання моделює фізичне розгортання артефактів на вузлах. Вузли зображені у вигляді прямокутних паралелепіпедів. Артефакти вузлів – як внутрішні прямокутники. Вузли можуть мати рівні вкладеності. Один вузол на такій діаграмі може представляти шаблон великої кількості фізичних вузлів, як-то кластер чи набір баз даних [13].

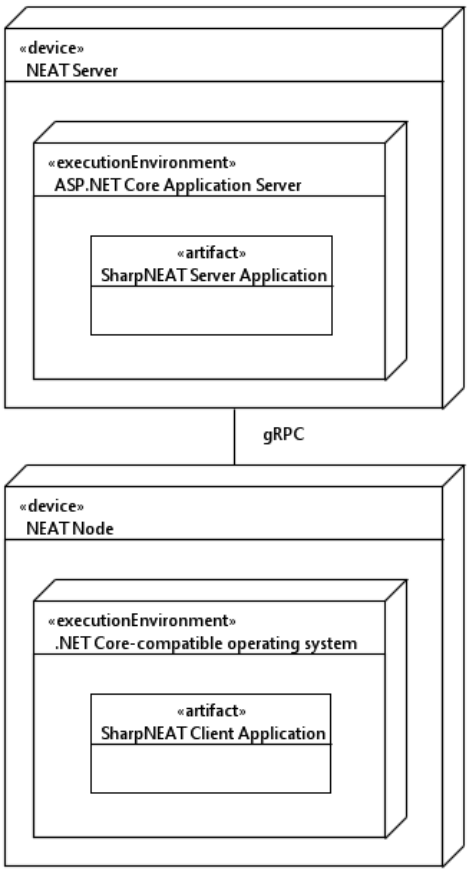


Рисунок 3. UML діаграма розгортання пропонованого рішення

Існує два типи вузлів: вузол-пристрій і вузол-середовище виконання. Вузли-пристрої – це фізичні обчислювальні ресурси, наприклад, комп'ютери та мобільні пристрої. Вузол середовища виконання – програмний обчислювальний ресурс, який виконують на батьківському вузлі, він надає сервіси й можливості для виконання іншого розгорнутого програмного забезпечення.

Наша реалізація включає два типи вузлів-пристроїв: NEAT Server та NEAT Client. NEAT Server – вузол-пристрій, що є головним сервером розподіленої реалізації методу нейроеволюції наростаючої топології. Цей сервер відповідає за виконання основної частини алгоритму NEAT, за управління та комунікацію з вузлами-клієнтами, а також за розподілення завдань. NEAT Client – вузол-пристрій, який є сукупністю пристроїв-клієнтів, відповідальних за частину оцінки придатності геномів в алгоритмі нейроеволюції наростаючої топології та за комунікацію з вузлом-сервером.

Наша реалізація включає два типи вузлів-середовищ виконання: ASP.NET Core Application Server та .NET Core-compatible operating system. Середовище виконання ASP.NET Core Application Server – .NET сумісна операційна система та реалізація ASP.NET Core серверу.

- ASP.NET Core – вільне та відкрите програмне забезпечення каркасу вебзастосунків, наступник ASP.NET від компанії Microsoft. ASP.NET Core є модульним фреймворком, що водночас підтримує виконання на операційній системі Windows з повноцінним фреймворком .NET Framework та на багатоплатформовій реалізації .NET [14].

- .NET Framework – програмний каркас, розроблений компанією Microsoft, що призначений переважно для виконання в середовищі сімейства операційних систем Microsoft Windows.

- .NET (попередньо .NET Core) – вільне та відкрите програмне забезпечення, програмний каркас для сімейств операційних систем Windows, Linux, macOS. Такий програмний каркас є багатоплатформовим наступником .NET Framework. Проєкт здебільшого підтримує та розробляє компанія Microsoft, випускають його під ліцензією MIT License.

Програмне забезпечення ASP.NET Core розгортається з такими вбудованими компонентами:

- Сервер Kestrel, багатоплатформна реалізація HTTP серверу. Kestrel надає найкращу доступну продуктивність виконання та використання пам'яті.

- Сервер IIS HTTP – сервер, що вбудований у головний процес для IIS.

IIS, Internet Information Services – вебсерверне програмне забезпечення, створене компанією Microsoft.

- Сервер HTTP.sys – ексклюзивний для сімейства операційних систем Windows HTTP сервер, що базується на драйвері ядра HTTP.sys та HTTP Server API.

На рис. 4 схематично зображено процес обробки мережевих запитів за участю компонентів Kestrel і коду цільового програмного забезпечення на базі ASP.NET Core:

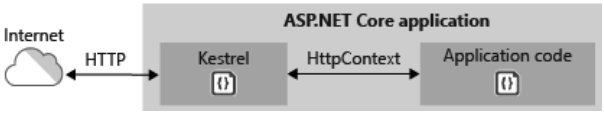


Рисунок 4. Схематичне зображення процесу обробки мережевих запитів

Ця реалізація включає два типи артефактів вузлів середовища виконання: SharpNEAT Server Application і SharpNEAT Client Application.

Артефакт SharpNEAT Server Application – це програмний застосунок на базі оригінального фреймворка SharpNEAT з модифікаціями й додатковими модулями для підтримки виконання в розподіленій системі. Ці модифікації та додаткові модулі відповідають за функції розподіленого виконання оцінки придатності геномів. Ця версія програмного застосунку виконує роль серверу – вона є центральним артефактом, відповідальним за виконання основної частини алгоритму NEAT, розподілення завдань між вузлами-клієнтами, комунікацію та управління ними.

Артефакт SharpNEAT Client Application також є програмним застосунком на базі оригінального фреймворка SharpNEAT із модифікаціями й додатковими модулями для підтримки виконання в розподіленій системі. Ця версія програмного застосунку виконує роль клієнта – вона є підпорядкованим артефактом-клієнтом, відповідальним за оцінку придатності отриманих завдань-геномів і комунікацію з вузлом-сервером.

На рис. 5 представлено UML діаграму послідовності взаємодії вказаних вузлів: NEAT Client і NEAT Server. Комунікація відбувається з використанням системи виклику віддалених процедур gRPC. Метод комунікації – потоковий, двосторонній. Порядок обробки повідомлень під час використання такого методу залежить від конкретної реалізації. Потоки незалежні: клієнт і сервер мають можливість надсилати та зчитувати повідомлення в будь-якому порядку. Наприклад, сервер може дочекатись отримання всіх повідомлень від клієнта перед надсиланням відповідей, або сервер і клієнт можуть почергово обмінюватися повідомленнями – сервер отримує запит, відповідає, отримує новий запит на основі відповіді й так далі.

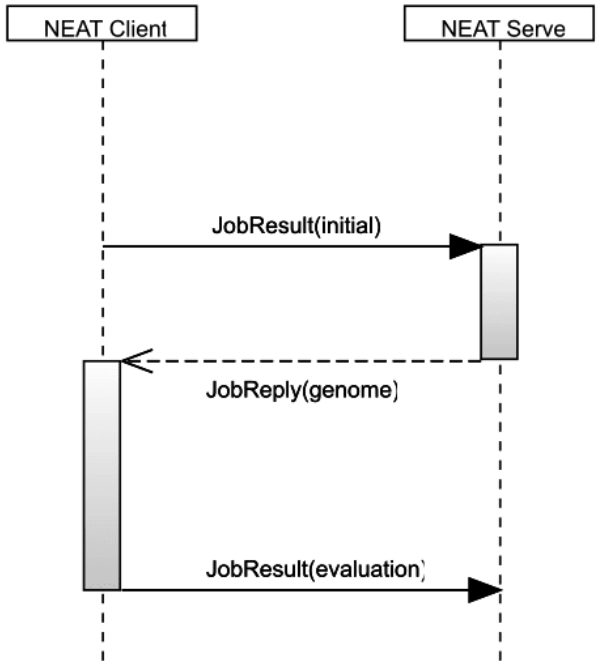


Рисунок 5. UML діаграма послідовності взаємодії вузлів

За такої реалізації існує два типи повідомлень: JobResult і JobReply. Комунікацію ініціює вузол-клієнт шляхом виклику RPC методу AcquireJob. Сервер отримує метадані клієнта та інформацію про метод. Параметри початкового повідомлення JobResult відсутні, тобто не містять ніяких результатів попередньої оцінки придатності. Сервер відповідає пакетом завдань-геномів JobReply за першої можливості. Після отримання завдання клієнт виконує оцінку придатності геномів і знову надсилає повідомлення JobResult, проте з даними оцінки

придатності. Після цього цикл JobReply (genome) – JobResult (evaluation) нескінченно повторюється.

Високорівневий огляд реалізації алгоритму нейроеволюції наростаючої топології у програмному фреймворку SharpNEAT був представлений на рис. 1. На рис. 6 зображено високорівневий огляд реалізації процедури оцінки чергового покоління. Саме процедура оцінювання виплоду є завданням розподілених обчислень нашої роботи.

Рисунок 6. Високорівневий огляд реалізації процедури оцінки чергового покоління

Виклик віддалених процедур

У сфері розподілених обчислень виклик віддалених процедур – це протокол, який дозволяє програмному забезпеченню викликати процедури (підпрограми) в іншому адресному просторі. Зазвичай під іншим адресним простором розуміють інший комп'ютер у спільній мережі. Однією з особливостей протоколу є відсутність необхідності розробки деталей комунікації. Більше того, такий протокол дозволяє реалізовувати міжпроцесорну комунікацію для стеків спочатку несумісних технологій програмного забезпечення. Виклик віддалених процедур – одна з форм клієнт-серверної взаємодії.

gRPC (Google Remote Procedure Calls) – вільна й відкрита програмна система виклику віддалених процедур. Система gRPC уперше була розроблена компанією Google. Вона використовує протокол HTTP/2 як транспорт, а Protocol Buffers (Protobuf) – як мову опису інтерфейсів. gRPC надає такі функції: аутентифікація, двостороння потокова комунікація, управління потоком передачі даних тощо. Існує багато реалізацій gRPC, які генерують сполучний код для легкої інтеграції в програмне забезпечення на цільових платформах і мовах [15].

Protobuf

Protocol Buffers – вільна й відкрита багатоплатформова бібліотека серіалізації структурованих даних. Бібліотека Protocol Buffers розроблена компанією Google. Структури даних у цій бібліотеці називають повідомленнями. Повідомлення і зв'язані сервіси описані в proto-файлі. Повідомлення серіалізуються в компактний, зворотно сумісний, бінарний, але такий, що не описує себе, формат [16].

Система gRPC використовує формат Protocol Buffers для серіалізації даних. На відміну від класичних прикладних програмних інтерфейсів HTTP JSON API, Protocol Buffers мають чітку єдину специфікацію, що дозволяє системі gRPC узгоджено працювати з даними між різними платформами та реалізаціями. Формат Protocol Buffers також пропонує високі показники продуктивності в питаннях серіалізації, десеріалізації і транспорту внаслідок бінарної природи формату повідомлень.

На наступному фрагменті зображено приклад Protocol Buffers специфікації сервісу й набору повідомлень розподіленої реалізації методу нейроеволюції наростаючої топології нашої роботи.

```
    service NeatProtoService {
       rpc AcquireJob (stream
JobResult)   returns   (stream
JobReply);
    }

    message JobResult {
      repeated GenomeTaskResult
genomeTaskResults = 1;
    }

    message JobReply {
         repeated  GenomeTask
genomeTasks = 1;
      string assemblyName = 2;
      string typeName = 3;
    }
```

```
message GenomeTask {
  string id = 1;
  repeated bytes genomes = 2;
}

message GenomeTaskResult {
  string id = 1;
  repeated double fitness = 2;
}
```

Схема має таку специфікацію сервісів та повідомлень:

- Повідомлення JobResult містить набір даних про виконану вузлом-клієнтом оцінку придатності.

- Повідомлення JobReply включає дані про завдання для клієнта, дані для конфігурації експерименту й алгоритму нейроеволюції наростаючої топології на клієнтській стороні та набір завдань GenomeTask.

- Повідомлення GenomeTask містить унікальний серверний ідентифікатор завдання і пакет серіалізованих геномів для оцінки придатності.

- Повідомлення GenomeTaskResult включає дані оцінки пакету геномів та унікальний серверний ідентифікатор завдання.

- Сервіс NeatProtoService містить специфікацію виклику віддалених процедур, а саме потоковий двосторонній метод AcquireJob, що визначає порядок використання повідомлень JobResult і JobReply.

Деталі реалізації

На рис. 7 зображено високорівневу діаграму модулів програмного застосунку серверної частини рішення.

- Модуль SharpNeat.GridServer відповідає за програмний сервер розподіленої реалізації методу нейроеволюції наростаючої топології. Цей модуль оркеструє виконання основної частини алгоритму NEAT, здійснює управління та комунікацію з вузлами-клієнтами, а також розподіляє завдання.

- Модуль SharpNeat.GridCommon містить спільні для програмних модулів SharpNeat.GridServer і SharpNeat.GridClient файли конфігурації головного завдання, порядок їхньої обробки під час складання проєкту, а також спільний proto-файл.

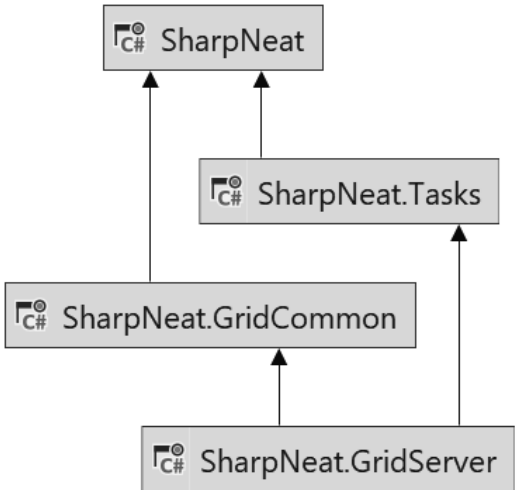


Рисунок 7. Високорівнева діаграма модулів програмного застосунку серверної частини рішення

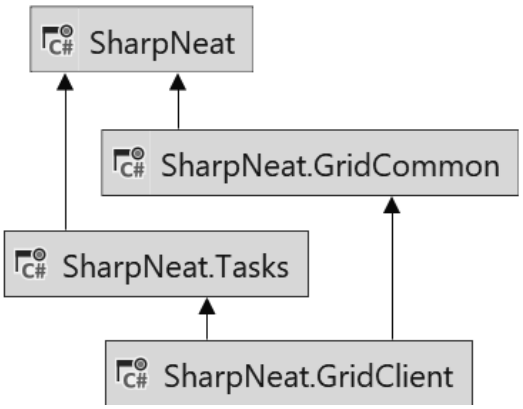


Рисунок 8. Високорівнева діаграма модулів програмного застосунку клієнтської частини рішення

На рис. 8 зображено високорівневу діаграму модулів програмного застосунку клієнтської частини рішення.

На рис. 9 зображено загальну високорівневу діаграму модулів програмного застосунку всього рішення.

Для забезпечення максимальної продуктивності розподілення завдань наша реалізація використовує такі особливості:

- патерн async/await;

- пакет System.Threading;

- пакет System.Threading.Tasks (Parallel, Task);

- пакет System.Collections.Concurrent (BlockingCollection, ConcurrentDictionary);
- пакет System.Linq.

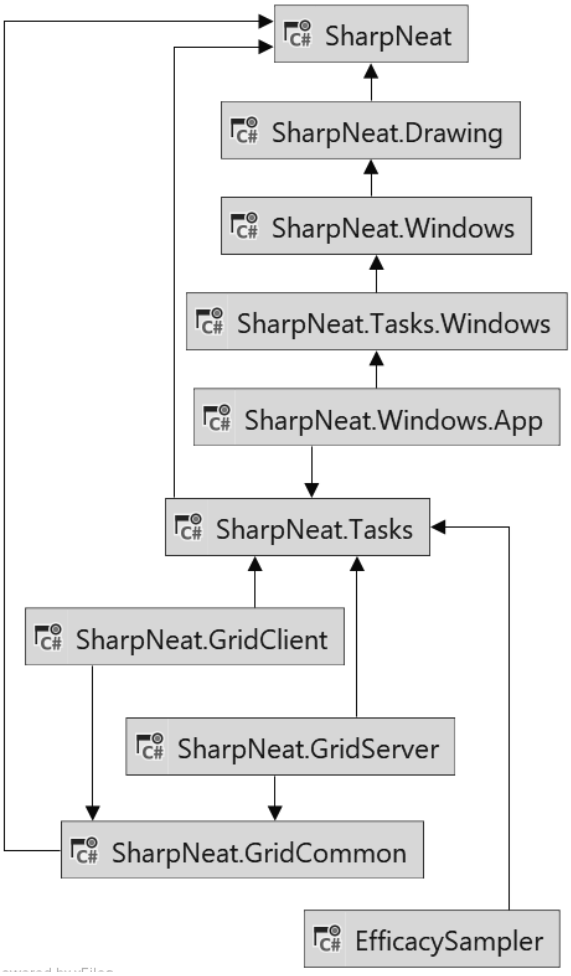


Рисунок 9. Загальна високорівнева діаграма модулів програмного застосунку рішення

Робота з вузлами й завданнями відбувається паралельно. Реалізація серверної частини є багатонитково-безпечною, що дозволяє уникати побічних негативних наслідків і непередбачуваної поведінки.

Бенчмаркінг

Як експеримент для бенчмаркінгу було реалізовано завдання двійкового мультиплексора на 20 входів (мультиплексор 16 в 1) [17]. На рис. 10 зображено умовну схему такого мультиплексора. Символом A позначені входи даних, символом B – адресні входи, Y – вихід. Нейронна мережа в цій задачі має 20 входів (20 нейронів вхідного шару), 4 з яких – адресні, а 16 – входи для даних. Усі входи приймають двійкові значення (0 або 1). У нейронної мережі є один вихід (один нейрон вихідного шару). Двійкова адреса подається на 4 адресні входи, що репрезентує вибір одного з 16 значень входів для даних. Оцінка складається з вичерпної перевірки нейронної мережі на кожній з 1048576 (2^{20}) можливих комбінацій входів. Вихідне значення нейронної мережі повинне збігатися зі значенням одного зі входів даних, що представлений двійковою адресою з адресних входів. Вихідне значення менше, ніж 0,5 уважають двійковим нулем, вихідне значення більше або рівне 0,5 – двійковою одиницею. Значення оцінки (придатності) адитивно розраховують у результаті ретельної перевірки. У разі відсутності помилок присуджують додаткову винагороду. Це завдання обране з огляду на одночасну простоту реалізації і складність виконання, що допоможе продемонструвати розподілення важких завдань між багатьма вузлами та їхню ефективну оркестрацію.

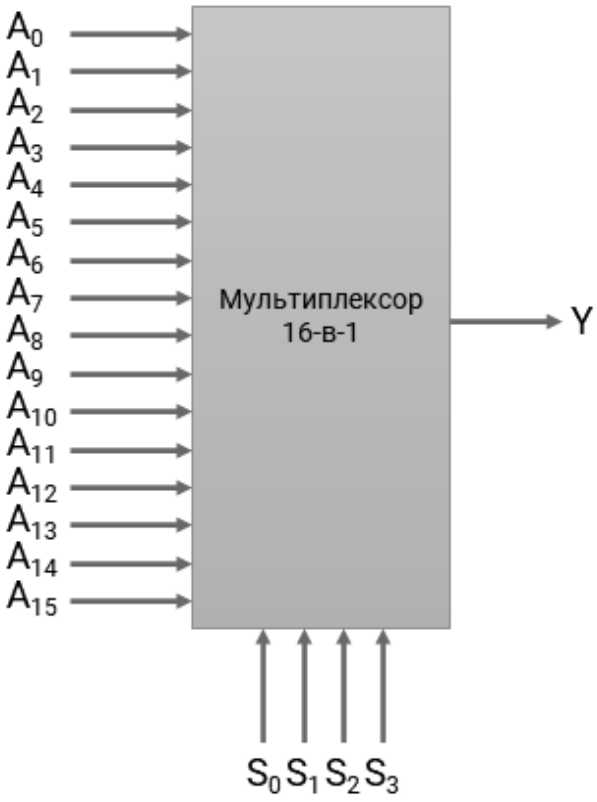


Рисунок 10. Умовна схема мультиплексора типу 16-в-1

Платформа хмарних обчислень Amazon Web Services пропонує широкий набір сервісів, можливостей і функцій для розгортання .NET додатків у хмарі AWS. Прикладами таких сервісів є контейнери Amazon Elastic Container Service (ECS) та Amazon Elastic Kubernetes Service (EKS), AWS Fargate, AWS Lambda, Amazon EC2 тощо [18].

Сімейства віртуальних машин на базі процесорів AWS Graviton2 пропонують вагомий виграш у продуктивності .NET додатків порівняно з еквівалентними віртуальними машинами сімейства Intel x86. Окрім цього, фреймворк .NET версії 5 включає ряд оптимізацій для архітектури ARM64.

Як обчислювальні вузли для бенчмаркінгу цього рішення використовувались віртуальні машини сервісу хмарних обчислювальних потужностей Amazon Elastic Compute Cloud (Amazon EC2) c6g.medium. Віртуальні машини Amazon EC2 C6g працюють на базі процесорів AWS Graviton2 з архітектурою ARM64. Віртуальні машини класу c6g.medium пропонують 1 віртуальний центральний процесор, 2 гігабайти оперативної пам'яті, до 10 Гбіт/с мережевої пропускної здатності та до 4750 Мбіт/с пропускної здатності сховища. Віртуальні машини такого типу зазвичай використовують для високопродуктивних обчислень, пакетної обробки даних, кодування відео, наукового моделювання, розподіленої аналітики, машинного навчання на центральних процесорах. Як вузол-сервер також може використовуватися одна віртуальна машина c6g.medium.

Задля створення умов середовища бенчмаркінгу, максимально наближених до реальних, вузли-клієнти й вузол-сервер були географічно віддалені на відстань приблизно в 500 кілометрів. Географічне розділення було досягнуто завдяки використанню AWS Availability Zone. Ця особливість допоможе відтворити реальну поведінку мережевої взаємодії вузлів, на відміну від використання високошвидкісних локальних мереж.

Ми використали таку конфігурацію віртуальних машин:

- операційна система: Ubuntu Server 20.04 LTS (HVM);
- накопичувач: EBS General Purpose (SSD);
- сімейство: c6g;
- кількість віртуальних центральних процесорів: 1;
- обсяг оперативної пам'яті: 2 Гб.

Для пакетного розгортання інфраструктури були використані функції Launch templates та AMIs. На рис. 11 зображено діаграму архітектури хмарного розгортання рішення на AWS.

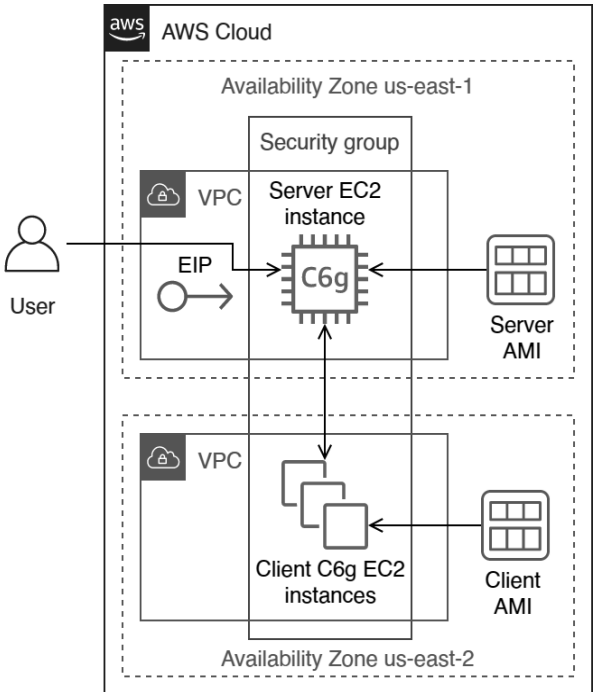


Рисунок 11. Діаграма архітектури хмарного розгортання рішення на AWS

Основні елементи інфраструктури:

- Availability Zone – ізольовані локації в різних географічних регіонах.
- Security group – віртуальний міжмережевий екран для контролю вхідного й вихідного трафіку.
- VPC (Amazon Virtual Private Cloud, Amazon VPC) – масштабована віртуальна приватна хмара, пул спільно використовуваних обчислювальних ресурсів.
- EIP (Elastic IP address) – статична, публічна IPv4 адреса, асоційована з віртуальною машиною.
- AMI (Amazon Machine Image) – підтримуваний образ операційної системи для використання в Amazon Elastic

Compute Cloud (Amazon EC2). AMI надає інформацію для запуску віртуальної машини.

- EC2 (Amazon Elastic Compute Cloud, Amazon EC2) – масштабовані обчислювальні ресурси в хмарі AWS.

З метою оптимізації продуктивності рішення, рівномірного розподілу завдань між вузлами та оптимального використання обчислювальних ресурсів була реалізована підтримка пакетної оцінки геномів. Тож, з'являється можливість зменшення накладних витрат обчислювальних ресурсів вузлів-клієнтів і вузла-сервера на частий обмін одиничними завданнями та їхніми результатами. Кількість завдань в одному пакеті була розрахована за такою формулою:

де: $size_{batch} = \frac{size_{population}}{size_{fleet}}$,

- $size_{batch}$ – кількість завдань в одному пакеті;
- $size_{population}$ – максимальна кількість завдань на оцінку одного покоління;
- $size_{fleet}$ – кількість обчислювальних вузлів-клієнтів.

Кількість завдань в одному пакеті в представленому рішенні конфігурується перед запуском серверного програмного додатка.

На рис. 12 зображено графік залежності швидкості оцінки від середньої кількості завдань у пакеті. Як видно з графіку, для подібного програмного рішення, середовища й завдання накладні витрати на взаємодію між вузлами нехтуються вже на найменших розмірах пакетів і дозволяють масштабувати розмір пакету з очікуваною продуктивністю. Для такого бенчмаркінгу використовувалися пакети розміром від 6 до 3000 завдань.

На рис. 13 зображено об'єднаний графік залежності середнього розміру пакету завдань і загальної швидкості оцінювання від розміру флоту вузлів-клієнтів.

На рис. 14 зображено графік залежності загальної швидкості оцінювання від розміру флоту вузлів-клієнтів.

Подані графіки засвідчують, що, використовуючи запропоноване розподілене рішення, швидкість виконання методу нейроеволюції наростаючої топології в частині оцінювання згенерованих нейронних мереж на прикладі розглянутого завдання зросла з 13 оцінювань за секунду до майже 3790 оцінювань за секунду при використанні 1024 хмарних обчислювальних вузлів зазначеної конфігурації.

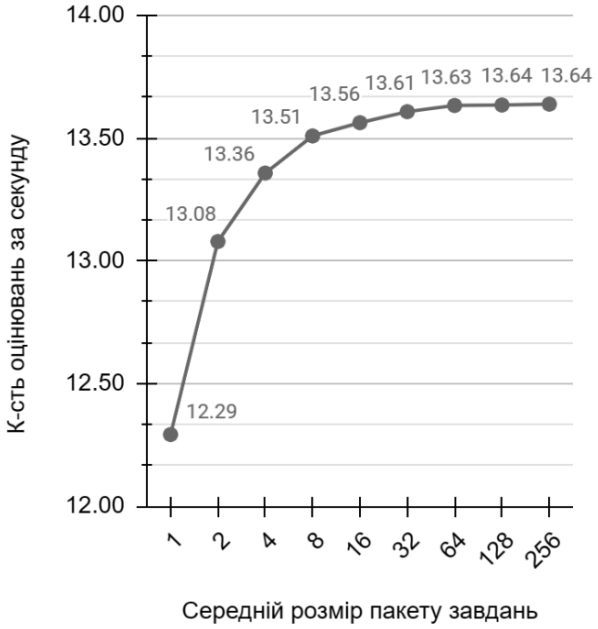


Рисунок 12. Графік залежності швидкості оцінки від середньої кількості завдань в пакеті

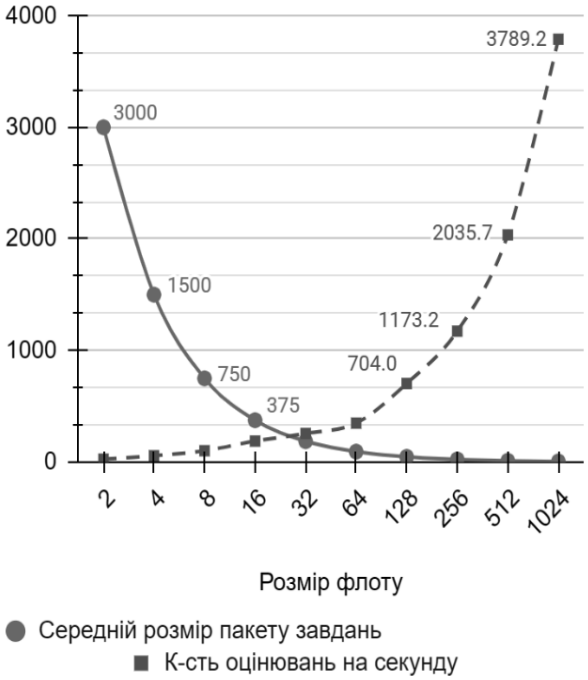


Рисунок 13. Об'єднаний графік залежності середнього розміру пакету завдань і загальної швидкості оцінювання від розміру флоту вузлів-клієнтів

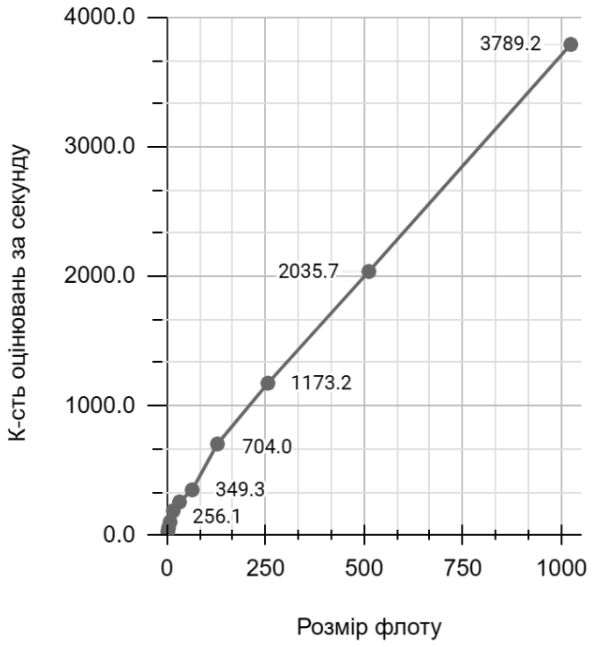


Рисунок 14. Графік залежності загальної швидкості оцінювання від розміру флоту вузлів-клієнтів

На рис. 15 зображено графік залежності швидкості оцінювання від порядкового номера покоління.

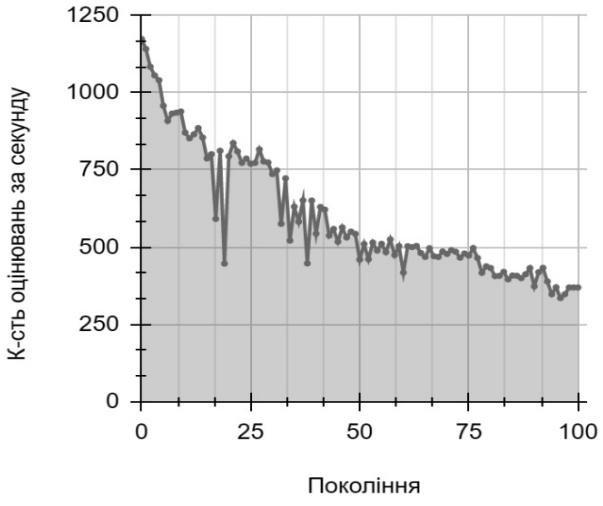


Рисунок 15. Графік залежності швидкості оцінювання від порядкового номера покоління.

Як видно з графіку, швидкість оцінювання зменшується з ходом еволюції. Це пов'язано з тим, що в процесі еволюції геноми (а, як наслідок, і представлені нейронні мережі) набувають об'ємніших і складніших топологій, тобто збільшується кількість шарів, нейронів і зв'язків між ними.

Висновки

У роботі запропоновано нову розподілену реалізацію методу нейроеволюції наростаючої топології, що за наявності достатніх обчислювальних ресурсів дозволяє радикально збільшити швидкість знаходження оптимальних конфігурацій нейронних мереж і допомагає обійти проблему повільної конвергенції до оптимальних результатів.

Література

1. Evolution 101: Neuroevolution | BEACON. BEACON | An NSF Center for the Study of Evolution in Action. URL: https://beacon-center.org/blog/2012/08/13/evolution-101-neuroevolution/ (дата звернення: 08.08.2021).
2. Субботін С. О., Олійник А. О., Олійник О. О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей : монографія / ред. Субботін С. О. Запоріжжя : ЗНТУ, 2009. 375 с.
3. Stanley K. O. Efficient evolution of neural networks through complexification : Thesis. 2004. URL: http://hdl.handle.net/2152/1266 (дата звернення: 08.08.2021).
4. NeuroEvolution of Augmenting Topologies. Department of Computer Science, College of Engineering and Computer Science@UCF. URL: http://www.cs.ucf.edu/~kstanley/neat.html (дата звернення: 08.08.2021).
5. Stanley K. O., Miikkulainen R. Evolving Neural Networks through Augmenting Topologies. Evolutionary Computation. 2002. Т. 10, № 2. С. 99–127. URL: https://doi.org/10.1162/106365602320169811 (дата звернення: 08.08.2021).
6. Stanley K. O., Bryant B. D., Miikkulainen R. Real-Time Neuroevolution in the NERO Video Game. IEEE Transactions on Evolutionary Computation. 2005. Т. 9, № 6. С. 653–668. URL: https://doi.org/10.1109/tevc.2005.856210 (дата звернення: 08.08.2021).
7. Stanley K. O., Miikkulainen R. Competitive Coevolution through Evolutionary Complexification. Journal of Artificial Intel-

ligence Research. 2004. T. 21. C. 63–100. URL: https://doi.org/10.1613/jair.1338 (дата звернення: 08.08.2021).
8. Green C. SharpNEAT Neuroevolution Framework. SharpNEAT Neuroevolution Framework. URL: https://sharpneat.sourceforge.io/ (дата звернення: 08.08.2021).
9. Andrews G. R. Foundations of multithreaded, parallel, and distributed programming. Reading, Mass : Addison-Wesley, 2000. 664 с.
10. Arora S. Computational complexity: A modern approach. Cambridge : Cambridge University Press, 2009.
11. Lynch N. A. Distributed algorithms. San Francisco, Calif : Morgan Kaufmann, 1997. 872 с.
12. Peleg D. Distributed computing: A locality-sensitive approach. Philadelphia : Society for Industrial and Applied Mathematics, 2000.
13. Booch G., Rumbaugh J., Jacobson I. Unified Modeling Language User Guide, The (2nd Edition) (The Addison-Wesley Object Technology Series). 2-ге вид. Addison-Wesley Professional, 2005. 496 с.
14. ASP.NET documentation. Developer tools, technical documentation and coding examples | Microsoft Docs. URL: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-5.0 (дата звернення: 08.08.2021).
15. Introduction to gRPC. gRPC. URL: https://grpc.io/docs/what-is-grpc/introduction/ (дата звернення: 08.08.2021).
16. Language Guide | Protocol Buffers | Google Developers. Google Developers. URL: https://developers.google.com/protocol-buffers/docs/overview (дата звернення: 08.08.2021).
17. The 11-multiplexer Problem. GEP: Home. URL: https://www.gene-expression-programming.com/webpapers/Ferreira-CS2001/Section6/SS5/SSS2.htm (дата звернення: 08.08.2021).
18. Powering .NET 5 with AWS Graviton2: Benchmarks | Amazon Web Services. Amazon Web Services. URL: https://aws.amazon.com/ru/blogs/compute/powering-net-5-with-aws-graviton2-benchmark-results/ (дата звернення: 08.08.2021).

References

1. Evolution 101: Neuroevolution | BEACON. BEACON | An NSF Center for the Study of Evolution in Action. URL: https://beacon-center.org/blog/2012/08/13/evolution-101-neuroevolution/ (date of access: 08.08.2021).
2. Subbotin S., Oliinyk A., Oliinyk O. Nonitterative, Evolutionary, and Multiagent Methods of Synthesis of Fuzzy Logic and Neural Network Models / ed. by S. O. Subbotin. Zaporizhzhya : ZNTU, 2009. 375 p.
3. Stanley K. O. Efficient evolution of neural networks through complexification : Thesis. 2004. URL: http://hdl.handle.net/2152/1266 (date of access: 08.08.2021).
4. NeuroEvolution of Augmenting Topologies. Department of Computer Science, College of Engineering and Computer Science@UCF. URL: http://www.cs.ucf.edu/~kstanley/neat.html (date of access: 08.08.2021).
5. Stanley K. O., Miikkulainen R. Evolving Neural Networks through Augmenting Topologies. Evolutionary Computation. 2002. Vol. 10, no. 2. P. 99–127. URL: https://doi.org/10.1162/106365602320169811 (date of access: 08.08.2021).
6. Stanley K. O., Bryant B. D., Miikkulainen R. Real-Time Neuroevolution in the NERO Video Game. IEEE Transactions on Evolutionary Computation. 2005. Vol. 9, no. 6. P. 653–668. URL: https://doi.org/10.1109/tevc.2005.856210 (date of access: 08.08.2021).
7. Stanley K. O., Miikkulainen R. Competitive Coevolution through Evolutionary Complexification. Journal of Artificial Intelligence Research. 2004. Vol. 21. P. 63–100. URL: https://doi.org/10.1613/jair.1338 (date of access: 08.08.2021).
8. Green C. SharpNEAT Neuroevolution Framework. SharpNEAT Neuroevolution Framework. URL: https://sharpneat.sourceforge.io/ (date of access: 08.08.2021).
9. Andrews G. R. Foundations of multithreaded, parallel, and distributed programming. Reading, Mass : Addison-Wesley, 2000. 664 p.
10. Arora S. Computational complexity: A modern approach. Cambridge : Cambridge University Press, 2009.
11. Lynch N. A. Distributed algorithms. San Francisco, Calif : Morgan Kaufmann, 1997. 872 p.

12. Peleg D. Distributed computing: A locality-sensitive approach. Philadelphia : Society for Industrial and Applied Mathematics, 2000.
13. Booch G., Rumbaugh J., Jacobson I. Unified Modeling Language User Guide, The (2nd Edition) (The Addison-Wesley Object Technology Series). 2nd ed. Addison-Wesley Professional, 2005. 496 p.
14. ASP.NET documentation. Developer tools, technical documentation and coding examples | Microsoft Docs. URL: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-5.0 (date of access: 08.08.2021).
15. Introduction to gRPC. gRPC. URL: https://grpc.io/docs/what-is-grpc/introduction/ (date of access: 08.08.2021).
16. Language Guide | Protocol Buffers | Google Developers. Google Developers. URL: https://developers.google.com/protocol-buffers/docs/overview (date of access: 08.08.2021).
17. The 11-multiplexer Problem. GEP: Home. URL: https://www.gene-expression-programming.com/webpapers/Ferreira-CS2001/Section6/SS5/SSS2.htm (date of access: 08.08.2021).
18. Powering .NET 5 with AWS Graviton2: Benchmarks | Amazon Web Services. Amazon Web Services. URL: https://aws.amazon.com/ru/blogs/compute/powering-net-5-with-aws-graviton2-benchmark-results/ (date of access: 08.08.2021).

Одержано 11.08.2021

***Про авторів*:**

Ашур Ілля Зін-Еддінович,
аспірант 3 курсу в
НТУУ “КПІ імені Ігоря Сікорського”,
https://orcid.org/0000-0003-2348-8777

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії
комп’ютерних обчислень,
професор кафедри інформаційних
систем та технологій НТУУ
“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

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: ilyaachour@gmail.com, doroshenkoanatoliy2@gmail.com

УДК 004.94 https://doi.org/10.15407/pp2021.03.016

О.В. Захарова

ВИЗНАЧЕННЯ СТУПЕНЯ СЕМАНТИЧНОЇ ПОДІБНОСТІ З ВИКОРИСТАННЯМ АПАРАТУ ДЕСКРИПТИВНИХ ЛОГІК

Встановлення семантичної подібності інформації є невід'ємою складовою процесу вирішення будь – яких задач інформаційного пошуку, в тому числі задач, пов'язаних з обробкою великих даних, виявленням семантичних веб сервісів, категоризації та класифікації інформації тощо. Введення спеціальних функцій для визначення кількісних показників ступеня семантичної відповідності інформації дозволяють ранжувати знайдену інформацію за її семантичною близькістю до цілі або пошукового запиту/ шаблону. Формування таких оцінок повинно враховувати багато аспектів: від сутності самих понять, семантична близькість яких підлягає оцінюванню, до особливостей бізнес-задачі, в межах вирішення якої це робиться. Зазвичай при побудові таких функцій подібності семантичні підходи поєднуються зі структурними, що забезпечують синтаксичне порівняння описів концептів. Це дозволяє суттєво деталізувати сам опис концепту, а вплив синтаксичної відповідності можна значно зменшити, використовуючи для представлення інформації виразніші дескриптивні логіки (ДЛ) та шляхом перенесення фокусу на семантичні властивості. ДЛ-онтології, на сьогодні, є найбільш розвинутим засобом представлення семантики, а механізми міркувань ДЛ забезпечують можливість логічного висновку. Більшість наведених у статті оцінок будуються на основі базових ДЛ, що підтримують лише констуктор перетину, але описані підходи можуть бути застосовані для будь-якої ДЛ, що забезпечує базові сервіси міркувань.

У роботі проведений аналіз існуючих підходів, моделей та способів оцінювання, заснованих на застосуванні апарату дескриптивних логік, запропонована їхня класифікація як за рівнем визначення подібності, так і за видами співставлення. Основна увага приділяється встановленню подібності концептів (моделям понятійного рівня). Задачі встановлення ступеня подібності між екземплярами та між концептом і екземпляром зводяться до знаходження найбільш специфічного концепту для екземпляра/екземплярів та оцінювання подібності відповідних концептів. Уведено поняття екзистенційної подібності та продемонстровано застосування певних видів оцінок для визначення ступеня семантичної подібності понять та/або знань на прикладі онтології геометричних понять.

Ключові слова: семантична подібність інформації, ступінь подібності концептів, найменше спільне покриття, оцінки вимірювання подібності, найбільш специфічний концепт, найбільш специфічний попередник, функція подібності, подібність за інформаційним змістом, семантична подібність за відповідністю ознак, функція відстані шляху, моделі оцінювання на основі властивостей, моделі оцінювання на основі семантичної мережі, моделі оцінювання на основі інформаційного контенту, екзистенційна подібність концептів, подібність екземплярів, подібність концепту та екземпляра, подібність ДЛ описів, GCS-подібність.

Вступ

Проблема знаходження семантично схожих понять та визначення ступеня їхньої подібності є нагальною як для вирішення прикладних задач (виявлення семантичних сервісів, ефективного семантичного пошуку інформації, категоризації даних тощо), так і більш загальних задач інформаційних технологій, як, наприклад, інтеграція онтологій, знань, інформаційний пошук тощо. Існує багато підходів, які намагаються вирішити проблеми визначення подібності методами текстового аналізу або за допомогою використання спеціальних словників понять, зокрема, словника Wordnet [1]. Водночас, зазвичай, розглядаються лише атомарні концепти, а складніші лишаються поза увагою. Окрім цього, опускаються варіанти виявлення подібності серед екземплярів понять та подібності екземпляра та концепту. Також, слід зазначити, що ступені інформаційної подібності повинні базуватися на семантиці, оскільки суто синтаксичний підхід є надто слабким, щоб забезпечити виконання стандартних виведень, особливо, якщо взяти до уваги виразні дескриптивні логіки (зокрема, $\mathcal{ALC}$), як засіб представлення знань. Зрозуміло, що алгоритми визначення та функції мір відповідності повинні бути ефективними. Якщо вони будуть надто складними, то навряд чи зможуть забезпечити потрібний результат у допустимий період часу та стати загально-використовуваними.

Останнім часом з'явилося чимало досліджень, у яких наголошується на доцільності

використання онтологій та функцій семантичної подібності на їхній основі для порівняння концептів та/або екземплярів концептів, котрі можна отримати через інтеграцію гетерогенних джерел інформації [2,3,4,5].

Види та рівні визначення подібності

Подібність інформації/знань можна розглядати та визначати на різних рівнях. А саме можна виділити:

1) понятійний рівнь – визначення подібності концептів;

2) рівень знань – визначення подібності екземплярів концептів;

3) змішаний рівень – визначення подібності концепту та екземпляра.

Оцінки, що вимірюють подібність, зазвичай використовують базову теорію множин та базуються на спільності об'єктів. Зокрема, базовий критерій для визначення таких мір можна сформулювати наступним чином: значення подібності між об'єктами є не лише результатом їх спільних характеристик, а й результатом їхніх відмінностей. Цей критерий відповідає теоретико-інформаційному визначенню подібності, а об'єктами в даному випадку є концепти та екземпляри концептів.

Розглянемо підходи до визначення можливих оцінок ступеня подібності та відповідні моделі оцінювання для кожного з перелічених рівнів. Але спочатку визначимо ряд понять, що використовуються більшістю існуючих моделей оцінювання.

Базові поняття та визначення

Визначення 1. **LCS (Least Concept Subsumer)** [23, 24] - найменше спільне покриття концептів. Нехай L дескриптивна логіка (ДЛ). Опис концепту E дескриптивної логіки $\mathcal{L}$ є найменшим загальним покриттям (LCS) описів концептів $C_1, \cdots, C_n$ в $\mathcal{L}$ (скорочено LCS($C_1, \cdots, C_n$)), якщо:

1) $C_i \sqsubseteq E$ для $i = 1, \cdots, n$ та

2) E є найменшим описом $\mathcal{L}$-концептом, що задовільняє (1), тобто, якщо E_0 є описом L-концепту такий, що $C_i \sqsubseteq E_0$ для всіх $i = 1, \cdots, n$, то $E \sqsubseteq E_0$.

Одразу слід зазначити, що LCS існує не для будь-якої ДЛ, що використовується для представлення знань. Але, якщо LCS існує, то воно є унікальним з точністю до еквівалентності. Всі міри, що розглядатимуться нижче, базуються на базовій ДЛ $\mathcal{ALC}$. У [6] показано, що для $\mathcal{ALC}$ логіки LCS існує завжди та задається диз'юнкцією понять. У разі, якщо диз'юнкція не підтримується логікою, LCS обчислюється вибором загальних імен понять в описах концептів (у межах понять універсуму та екзистенційних обмежень для тієї самої ролі), не зважаючи на TBox в цілому [6]. Але в такому разі результат обчислення LCS може бути надто загальним. За цими міркуваннями LCS обчислюється відносно TBox, на основі якого визначаються концепти [7].

Беручи до уваги TBox, визначення LCS можна переформулювати так.

Визначення 2. Нехай $\mathcal{L}1$ та $\mathcal{L}2$ дескриптивні логіки такі, що $\mathcal{L}1$ є під-логікою $\mathcal{L}2$, тобто $\mathcal{L}1$ містить менше конструкторів, які використовуються для побудови виразів. Для заданого TBox логіки $\mathcal{L}2$ $\mathcal{T}$, $\mathcal{L}1(\mathcal{T})$ – множина описів концептів, які можуть містити концепти, що визначені у $\mathcal{T}$. $C_1, \ldots, C_n$ описи концептів з $\mathcal{L}1(\mathcal{T})$, тоді LCS($C_1, \ldots, C_n$) у $\mathcal{L}1(\mathcal{T})$ відносно TBox $\mathcal{T}$ є опис найбільш специфічного $\mathcal{L}1(\mathcal{T})$ концепту, який включає $C_1, \ldots, C_n$ на TBox $\mathcal{T}$, а саме, це такий опис $\mathcal{L}1(\mathcal{T})$- концепту D, що:

3) C_i D для $i = 1, \cdots, n$ та

4) Якщо E є описом $\mathcal{L}1(\mathcal{T})$- концепту, що задовільняє C_i E для всіх $i = 1, \cdots, n$, то $D \sqsubseteq E$

Якщо LCS відповідно для TBox не існує (наприклад, у разі циклічного TBox), обчислюється його апроксимація, що називається Гарне Загальне Покриття (GCS) [25] відносно TBox та існує навіть для загального TBox. GCS обчислюється через визначення найменшої кон'юнкції концептів та їхніх заперечень, що може включати кон'юнкцію імен концептів вищого рівня для кожного з концептів, що розглядається, та аналогічної кон'юнкції концептів, які становлять ранг екзистенційних та універсальних обмежень відносно тієї самої ролі. GCS є специфічнішим покриттям, ніж LCS, що обчислюється безвідносно Tbox. Хоча у загальному випадку воно включає (або є еквівалентним) LCS, що обчислюється відносно TBox [7].

MSA (Most Specific is-a Ancestor) [8] – найбільш специфічний попередник в ієрархії таксономії. Дане поняття визначається як бінарне відношення на таксономії концептів, але за семантикою воно подібне до LCS. Ці обидва поняття обчислюють найспецифічніше уза-

гальненя вхідних концептів (відносно операції включення). Різниця ж полягає в тому, що MSA працює на таксономії понять та повертає один концепт, який містить два вихідних концепти (є їхнім is-a попередником) та не включає жодного іншого, який би задовільняв ті ж вимоги. A LCS є описом, який покриває вихідні концепти, та, як результат, при його обчисленні повертаються всі включені до нього концепти. Якщо концепти зв'язані лише родо-видовими зв'язками, тобто TBox є таксономією, LCS покриття концептів вироджується до одного попередника і $LCS(C_1,C_2)=MSA(C_1,C_2)$.

MSC - найбільш специфічний концепт. Унарне відношення на множині екземплярів ABox.

Визначення 3. [25] Нехай дано ABox $\mathcal{A}$ та екземпляр α цього ABox, найбільш специфічним концептом для екземпляра a відносно ABox $\mathcal{A}$ є концепт C, позначається , такий, що $\mathcal{A}C()$, та такого, що $\mathcal{A}\models D()$, $C\sqsubseteq D$ (де $\models$ позначає оператор виведення).

Одразу слід зазначити, що в загальному випадку ациклічного ABox у виразній ДЛ не може бути виражений кінцевим описом концепту [2], можна отримати лише його апроксимацію. Тож, існування найбільш специфічного концепту для індивіда ABox не є гарантованим, або його складно обчислити, й апроксимацію обмежують певною встановленою глибиною. Максимальна глибина апроксимації, як визначено у [20], відповідає глибині ABox. У такому разі, для будь - якого екземпляра ABox α можемо визначити найбільш специфічний концепт MSC(α) або його апроксимацію MSC*(α).

Визначення семантичної подібності концептів

На сьогодні існує чимало досліджень, автори яких намагаються перевести семантичні відношення між поняттями у деякі кількісні показники. Зрозуміло, що на принципи формування таких оцінок впливає, насамперед, сутність самих понять, семантична близькість яких оцінюється, а також задача, для вирішення якої обираються чи визначаються функції подібності. Більшість існуючих досліджень застосовують семантичний підхід у поєднанні зі структурним, який порівнює описи концептів, що розглядаються. Звісно, це дозволяє суттєво деталізувати опис, а вплив синтаксичної відповідності можна значно зменшити при використанні для представлення інформації більш виразних ДЛ та перенесення фокусу на семантичні властивості.

При встановленні ступеня семантичної відповідності між концептами однієї онтології функція подібності фактично є відображенням $\mathcal{S}$: $\mathcal{L}(\mathcal{T})\times\mathcal{L}(\mathcal{T})\rightarrow Y$, де $\mathcal{T}$ є TBox даної онтології, представлений у ДЛ $\mathcal{L}$, а Y є дійсним числом, що кількісно визначає ступінь подібності. В оцінках, що базуються на співвідношеннях (частках), $Y\in[0,1]$, але існують й інші моделі вимірювань.

У загальному випадку задача суттєво ускладнюється. Якщо відповідність встановлюється між концептами двох різних онтологій з TBox-амі $\mathcal{T}1$ та $\mathcal{T}2$ ДЛ, відповідно $\mathcal{L}1$ та $\mathcal{L}2$, необхідно побудувати відображення $\mathcal{S}$: $\mathcal{L}1(\mathcal{T}1)\times\mathcal{L}2(\mathcal{T}2)\rightarrow Y$.

У будь-якому випадку функція повинна мати наступні властивості:

1) Нехай E – певна множина елементів (об'єктів однієї чи різних онтологій), для яких визначається ступінь подібності, то функція $\mathcal{S}$ визначена на множині $E\times E$

2) Функція $\mathcal{S}$ є позитивно-визначеною, тобто $\mathcal{S}(C,D)\geq 0$

3) $\forall C,D: \mathcal{S}(C,D)\leq\mathcal{S}(C,C)$

При визначенні функції відповідності, необхідно розуміти, що подібність концептів можна розглядати як з боку ступеня їх подібності, так і ступеня їх відмінності. І функція подібності повинна мати позитивну кореляцію зі ступенем подібності між концептами та негативну - з показником відмінності між ними. Зрозуміло, що цей числовий показник залежить від багатьох факторів, а саме: специфіки досліджуваного контенту, виразності та однорідності мов представлення онтологій тощо. Але ключовим питанням при створенні функції подібності є «як виміряти ступінь подібності (відмінності) концептів». Це, в свою чергу, пов'язано з тим, як збирається досліджувана інформація. Навряд чи показник подібності можна розцінювати, як абсолютну оцінку, але він має забезпечувати можливість достовірного ранжування концептів за ступенем їхньої подібності. Серед основних підходів до побудови такої функції можна виділити:

1) визначення подібності як функції відстані шляху між таксонами в ієрархії, що лежить в основі цієї онтології [10, 11, 12];

2) оцінка семантичної подібності за відповідністю ознак [13];

3) визначення ступеня подібності концептів за інформаційним змістом [14,15];

4) екзістенціональна подібність понять.

Перший підхід може бути застосований лише в межах однієї онтології, тобто його використання може бути доцільним лише, якщо оцінювання виконується на базі одного джерела інформації, і досліджувані поняття, є концептами однієї онтології або інтегрованої онтології вихідних джерел інформації. Другий підхід для обчислення семантичної подібності використовує як загальні, так і дискримінантні ознаки між поняттями та / або екземплярами понять. Методи третьої групи засновані на теорії інформації. Вони визначають ступінь подібності між двома поняттями в рамках ієрархії понять з точки зору кількості інформації, що передається безпосередньо супер-концептом, який включає порівнювані концепти. Всі оцінки, які базуються на ознаках та властивостях концептів, можна назвати оцінками інтенсіональної подібності. Під *екзистенційною подібністю понять* будемо розуміти ступінь їхньої близькості за множинами екземплярів, які вони містять.

У разі встановлення ступеня відповідності між поняттями різних, можливо, різнорідних, онтологій, перелічені вище підходи працюють за умови виконання певних чинників та обмежень. По-перше, формальне представлення цих онтологій повинна підтримувати механізми міркувань, такі як включення. (Слід одразу зазначити, що механізм включення підтримується базовими ДЛ такими, зокрема, як $\mathcal{ALC}$). По-друге, застосування підходів оцінювання базується на використанні узагальненої онтології, а локальні концепти в різних онтологіях повинні успадковувати структуру визначення з їх узагальненої онтології. У [16] пропонуються деякі підходи до порівняння таких концептів із різних онтологій за складом їхніх екземплярів. А саме, робиться припущення, що при виконанні визначених вище обмежень, ознакою відповідності двох концептів може бути перетин множин їхніх екземплярів. А для порівняння описів понять, які можна поєднати у загальну онтологію, використовуються три основні підходи, а саме:

- фільтрація на основі відстані-шляху між поняттями у загальній онтології;
- визначення ступеня подібності на основі відповідності елементів графів (один до одного) описів понять;
- визначення ймовірносних метрик, які визначають подібність з точки зору спільного розподілу понять.

Також, якщо обчислення оцінок подібності робиться для концептів з різних онтологій, необхідно враховувати різницю між рівнями формалізації специфікацій цих онтологій. Зокрема, у [17] функція відповідності визначає класи подібних сутностей за допомогою співставлення з використанням наборів синонімів, семантичного сусідства та дискримінаційних ознак, що класифіковані за частинами, функціями та атрибутами. У [9] представлений інший підхід, спрямований на знаходження спільних властивостей серед концептів або тверджень.

Перелічені групи підходів до оцінювання подібності базуються на відповідних моделях.

Основні моделі оцінювання

До найбільш розповсюджених моделей оцінювання можна віднести:

- моделі на основі властивостей;
- моделі на основі семантичних мереж;
- моделі на основі інформаційного контенту.

В **моделях на основі властивостей концепт** C характеризується множиною своїх властивостей, що позначається ftrs(C). У [18] пропонується дві групи вимірювань для такої моделі:

1) контрастна модель, де подібність двох концептів C і D визначається лінійною функцією

$$contra(C,D) = \theta f\big(ftrs(C) \cap ftrs(D)\big) - \alpha f(ftrs(C) \backslash ftrs(D)) - \beta f(ftrs(D) \backslash ftrs(C)),$$

де \ - операція різниці множин, α, β та θ не негативні константи, а f(.) – виражає кількість ознак в множині

2) нормалізована модель співвідношення, де подібність визначається як частка множин:

$$sim(C,D) =_{def} \frac{f(ftrs(C) \cap ftrs(D))}{f\big(ftrs(C) \cap ftrs(D)\big) + \alpha f\big(ftrs(C) \backslash ftrs(D)\big) + \beta f(ftrs(D) \backslash ftrs(C))}$$

Якщо вважати, що функція подібності є симетричною, то Якщо припустити, що функція є дистрибутивною по множинам, що не перетинаються, можна перетворити наступним чином:

$$sim(C,D) =_{def} \frac{2f(ftrs(C) \cap ftrs(D))}{f(ftrs(C)) + f(ftrs(D))}$$

У **моделях, що засновані на семантичній мережі**, довідкова інформація надається у формі семантичної мережі, що включає концепти та, принаймні, is-a ребра (іноді розглядаються більш складні відносини, як у WordNet). Це є прикладом саме того випадку, коли оцінювання подібності базується на вимірюванні довжини шляху між концептами у мережі. Якщо концепти знаходяться у таксономії, тобто пов'язані родовидовими відношеннями, значення подібності між двома концептами обчислюються кількістю ребер на шляху від концептів, що розглядаються, до їх найближчого попередника. Якщо поняття розділені лише декількома зв'язками, то вони вважаються подібними. Чим більше зв'язків їх розділяють, тим менша схожість між ними [8, 19, 12, 20]. Тобто, для оцінювання відповідності концептів C і D знаходиться найбільш специфічний is-a попередник E = MSA(C,D) концептів C і D та обчислюється міра подібності як сума довжин шляхів від C до E та від E до D. Розвинутіші оцінки можуть враховувати глибину концепту MSA(C, D), щільність ребер у вузлах шляху та вагу ребер.

У **моделях, заснованих на інформаційному контенті**, разом із семантичною мережею використовується інформація про імовірність того, що сутність описується конкретним концептом C. Така імовірність зазвичай оцінюється на основі вихідної специфічної задачі.

Величина інформаційного контенту концепту вимірюється на основі імовірності $pr(\mathrm{C})$, як $IC(\mathrm{C}) =_{def} -\log pr(\mathrm{C})$. У [21] пропонується міра подібності концептів C та D на основі імовірносної оцінки їх MSA:

$$sim(C,D) =_{def} IC(MSA(C,D)) =_{def} -\log pr(MSA(C,D)).$$

У [22] пропонується міра відстані концептів у мережі на основі їхнього інформаційного контенту, що враховує такі фактори, як глибина та щільність ребер шляху між концептами:

$$dist(C,D) =_{def} IC(C) + IC(D) - 2IC(MSA(C,D))$$

У [18] пропонується міра подібності, що визначається часткою:

$$sim(C,D) =_{def} \frac{2IC(MSA(C,D))}{IC(C) + IC(D)}$$

Визначення оцінювання подібності для ДЛ описів

Усі метрики, представлені вище, визначені на атомарних концептах. Але наведені оцінки можна переформулювати й для складних концептів, які визначаються через атомарні засобами ДЛ. Зазначимо, що при побудові оцінок ми вважаємо, що описи концептів представлені у базовій ДЛ, яка підтримує лише операцію перетину концептів. Будь-який опис концепту можна привести до його нормальної форми, тобто розкласти так, щоб він містив лише атомарні концепти. Зазвичай, це робиться просто шляхом підстановки у визначеня замість неатомарних концептів їхні описи. Позначимо через nf(C) множину атомарних концептів, що зустрічаються у нормальній формі концепта C. Зазначимо, що C⊑D (де ⊑ - просте структурне включення), якщо nf(D) ⊆ nf(C).

Із урахуванням наведеного визначення струкурного опису концепту можна переформулювати наведені вище оцінки відповідності наступним чином.

Для моделі властивостей будемо розглядати властивості концепту як атомарні концепти, а складний концепт як кон'юнкцію цих атомарних концептів. Враховуючи особливості перетину та різниці множин атомарних властивостей, міри відповідності, за умови їхньої симетричності, можна визначити так:

$$contra(C,D) =_{def} f(lcs(C,D)) - 0{,}5 * * f(diff(C,D)) - 0{,}5 * f(diff(D,C))$$

$$sim(C,D) =_{def} \frac{2 * f((C,D))}{2 * f(lsc(C,D)) + f(diff(C,D) + f(diff(D,C))}$$

Слід зазначити, що у наведених мірах функція f є лічильником властивостей, можливо зважених.

Наразі розглянемо модель семантичної мережі. Якщо мережа є ієрархією, та

концепт С має is-a попередників: U_1, U_2, U_3, …, U_{n-1}, Un, введемо концепт С* такий, що $C := C^*U_1U_2U_3...U_{n-1}U_n$. У результуючому Т-Box, концепт, що визначається, має таку саме ієрархію включення як вихідні вузли у семантичній мережі, більше того, якщо у вихідній мережі шлях U_1, U_2, . . . U_n = до кореня is-a ієрархії, то нормальна форма концепта U_1 у ДЛ $nf(U_1)$ = . Іншими словами, якщо мережа є деревом, то кардинальність концепту, що є нормальною формою концепту С $|nf(C)|$ дорівнює довжині шляху від U_1 до кореня. Шляхи від концептів С та D до кореня перетинаються в $E=MSA(C,D)$, що на ієрархії включення співпадає з $LCS(C,D)$. Тоді відстань між концептами С і D можна визначити, як:

$$dist(C,D) =_{def} |nf(C)| + |nf(D)| - 2 * |nf(LCS(C,D))|,$$ де |X| - кардинальність концепту X.

Відповідно для інформаційних моделей:

$$dist(C,D) =_{def} IC(C) + IC(D) - 2 * IC(lcs(C,D)),$$

$$sim(C,D) =_{def} \frac{2 * IC(lsc(C,D))}{IC(C) + IC(D)}$$

Екзистенційні оцінки подібності концептів

В екзистенційних підходах значення подібності обчислюється шляхом підрахунку спільних екземплярів розширень концептів [26] або шляхом вимірювання варіації змісту між концептами, що розглядається у [27, 28, 29].

Зазвичай, онтологія має структуру, складнішу за просту таксономію, і оцінки подібності, що базуються на відстанях в таксономії або на використанні поняття найбільш специфічного попередника (MSA), використовуватися не можуть.

Слід зазначити, що семантичне відношення включення базується на канонічній інтерпретації ABox дескриптивної логіки та припущенні унікального простору імен (UNA), з якої випливає, що інтерпретацією екземплярів ABox є вони самі, а також індивіди, що відповідають різним об'єктам предметної області, мають різні імена в просторі імен. Тому визначимо оцінку відповідності концептів на основі їх розширення у каноничній інтепретації ДЛ [25].

Нехай – множина концептів у ДЛ $\mathcal{ALC}$, а $\mathcal{A}$ – ABox з каноничною інтерпретацією $\mathcal{I}$. Семантична подібність концептів $\mathcal{s}$ є функцією: $\mathcal{s}$:$\mathcal{L}\times\mathcal{L}\rightarrow$[0,1], що визначається наступним чином:

$$\mathcal{s}(C,D) = \frac{|I^{\mathcal{I}}|}{|C^{\mathcal{I}}|+|D^{\mathcal{I}}|-|I^{\mathcal{I}}|} * \max(|I^{\mathcal{I}}|/|C^{\mathcal{I}}|, |I^{\mathcal{I}}|/|D^{\mathcal{I}}|),$$ де $I = C \sqcap D$ та $(.)^{\mathcal{I}}$ розширення концепта в інтерпретації $\mathcal{I}$.

Наведену оцінку можна обгрунтувати так. Якщо концепти С та D еквівалентні, тобто C⊑D та D⊑C, $\mathcal{s}$=1. Якщо концепти є взагалі різними, та їхні розширення не перетинаються, оцінка є мінімальною, тобто її значення дорівнює 0. У випадку *непустого перетину концептів оцінка набуває* значення в діапазоні від 0 до 1. Тобто дана оцінка виражає ступінь подібності концептів С та D, зменшену на $\max(|I^{\mathcal{I}}|/|C^{\mathcal{I}}|, |I^{\mathcal{I}}|/|D^{\mathcal{I}}|)$, що, у свою чергу, представляє несхожість цих концептів. Це означає, що ступінь подібності розглядається не як абсолютна величина, а як зважена щодо ступеня несхожесті. Така оцінка відповідає досить міцному семантичному зв'язку між поняттями, що забезпечується відношенням включення.

Оцінки GCS-подібності концептів

Міри GCS-подібності визначаються на основі поняття GCS-покриття та можуть бути застосовані, якщо оцінки, які базуються на перекритті розширень концептів, інформаційному контенті чи на відстанях між концептами, не працюють. Оцінки на основі GCS також використовують поняття розширення концепту, але замість підрахунку спільних екземплярів даних концептів, значення подібності визначається як варіація числа екземплярів у розширенні концепту відносно числа екземплярів у розширенні їх загального супер-концепту. Загальний супер-концепт визначається через GCS концептів, а оцінка відносно TBox $\mathcal{T}$ логіки $\mathcal{ALC}$ формально визначається таким чином.

$\mathcal{T}$ - $\mathcal{ALC}$- TBox. $\mathcal{L}$ – дескриптивна логіка, що включає $\mathcal{ALC}$. С і D описи концептів в $\mathcal{L}(\mathcal{T})$. Тоді міра семантичної подібності $\mathcal{S}$ є функцією $\mathcal{S}$: $\mathcal{L}(\mathcal{T}) \times \mathcal{L}(\mathcal{T}) \rightarrow$[0,1], що визначається так:

$$\mathcal{S}(\mathrm{C},D)=\frac{\min\left(|C^{\mathcal{J}}|,|D^{\mathcal{J}}|\right)}{|(GCS(C,D))^{\mathcal{J}}|}*$$

$$*\left(1-\frac{\left|(GCS(C,D))^{\mathcal{J}}\right|}{|\Delta^{\mathcal{J}}|}*\left(1-\frac{\min(|C^{\mathcal{J}}|,|D^{\mathcal{J}}|)}{\left|(GCS(C,D))^{\mathcal{J}}\right|}\right)\right),$$

де – обчислює розширення концепту відносно інтерпретації $\mathcal{J}$ (канонічної інтерпретації [2, 9]).

Тобто, якщо два концепти семантично подібні, вони повинні мати гарний спільний супер-концепт, що є близьким до обох концептів, а саме розширення суперконцепту, що містить багато екземплярів, спільних з вихідними концептами. В такому разі значення функції буде наближатися до одиниці. Навпаки, якщо вихідні концепти дуже різні, то їхні GCS та суперконцепт містить багато екземплярів, що не належать вихідним концептам, тобто значення оцінки подібності буде наближатися до 0. Дана міра не вимагає перетину концептів, що розглядаються, та не бере до уваги відстань семантичного шляху. Більше того, щоб запобігти отриманню некоректного значення подібності тоді, коли один концепт є дуже подібним до супер-концепту та дуже відрізняється від іншого концепту, що порівнюється, у визначенні міри розглядається мінімальне розширення концептів.

Визначення оцінок подібності на рівні знань та змішаному рівні

Нагадаємо, що до оцінок цих рівнів відносяться оцінки визначення ступеня подібності екземплярів та екземпляра і концепту. Визначення міри із залученням екземплярів базується на понятті Найбільш Специфічного Концепту (MSC). Для кожного екземпляра в ABox можна обчислити MSC або його апроксимацію. В деяких випадках ці поняття є еквівалентними.

Нехай *a* і *b* – два екземпляри ABox, *A*=MSC*(a), B*=MSC*(b)*. Тоді міри семантичної подібності можуть бути застосовані до описів концептів *A** та *B**, й результуюча оцінка виражатиме ступінь подібності відповідних екземплярів:

$$\forall a,b\colon s(a,b)=s(A^*,B^*)=s(MSC^*(a),MSC^*(b))$$

Аналогічно, значення подібності між описом концепту C та екземпляра a може бути обчислене шляхом визначення апроксимації MSC екземпляра та наступного застосування міри подібності до концепта C та апроксимації MSC* екземпляра a:

$$\forall a,C\colon s(a,C)=s(A^*,C)=s(MSC^*(a),C)$$

Тож, обидві оцінки зводяться до визначення подібності описів концептів після попередньої апроксимації екземплярів. Одночасно можуть бути використані будь-які наведені вище моделі обчислення ступеня відповідності концептів.

Слід зазначити, що складність запропонованих методів залежить від складності стандартних методів виведення в ДЛ.

Застосування оцінок подібності на прикладі ДЛ онтології POGeometry

Розглянемо застосування міри відповідності описів концептів на основі їхнього розширення у канонічній інтерпретації ДЛ на прикладі онтології домену геометричних понять *POGeometry*.

TBox онтології домену *POGeometry:*

Coordinate, GeometricFigure

Vertex ⊑ *has.XCoordinate*

Vertex ⊑ has.*YCoordinate*

XCoordinate ⊑ *Coordinate*

YCoordinate ⊑ *Coordinate*

Coordinate ⊑ *hasValue.NUMBER*

Vector ⊑ *2has.Vertex*

Vector ⊑ h*as.VectorLength*

Vector ⊑ *has.VectorAngle*

VerterxLength ⊑ *hasType.NUMBER*

VerterxAngle ⊑ *hasType.NUMBER*

Height ⊑ *hasType.NUMBER*

EdgeLenth ⊑ *hasType.NUMBER*

...

Polygon ≡ *GeometricFigure* ⊓=*has.Vertex*⊓ =*has.Vector*

Circle ⊑ *GeometricFigure*

Quadrangle ≡ *Polygon* ⊓=*4has.Vertex*⊓ =*4has.Vector*

Triangle ≡ *Polygon* ⊓ =*3has.Vertex* =*3has.Vector*

Polygon ⊑ *has.Vertex*

Polygon ⊑ *has.Vector*

Triangle ⊑ =*3has.Height*

Square ⊑ *hasType.NUMBER*

GeometricFigure ⊑ *has.Square*

Circle ⊑ *GeometricFigure*

ABox:

Triangle(ABC), Triangle(XYZ), Triangle(A1B1C1), Triangle(B1C1D1), Triangle(A1C1D1), Triangle(A1B1D1), Triangle(X1X2X3), Triangle(X2X3X4), Triangle(X3X4X5), Triangle(X4X5X6), ..., Quadrangle(A1B1C1D1), Polygon(X1X2X3X4X5X6.), Circle(O_1), Circle(O_2)

Враховуючи визначення концептів *Quadrangle* та *Triangle* можна вивести включення концептів *Triangle* ⊑ *Polygon* та *Quadrangle* ⊑ *Polygon*. Отже, всі екземпляри концептів *Triangle* та *Quadrangle* є екземплярами концепту *Polygon*.

Тобто $|Polygon^{\mathcal{I}}|$=47, $|Triangle^{\mathcal{I}}|$=29, $|Quadrangle^{\mathcal{I}}|$=17.

Тоді відповідність концептів *Triangle* та *Polygon* можна визначити на основі їхніх множин екземплярів таким способом:

Нехай $I = Triangle \sqcap Polygon,$, тоді

$$s(Polygon, Triangle) = \\ = \frac{|I^{\mathcal{I}}|}{|Polygon^{\mathcal{I}}| + |Triangle^{\mathcal{I}}| - |I^{\mathcal{I}}|} * \\ * \max\left(\frac{|I^{\mathcal{I}}|}{|Polygon^{\mathcal{I}}|}, \frac{|I^{\mathcal{I}}|}{|Triangle^{\mathcal{I}}|}\right) \\ = \frac{29}{47 + 29 - 29} * \max\left(\frac{29}{47}, \frac{29}{29}\right) = \frac{29}{47} = 0{,}62$$

Зважаючи на те, що інтерпретації концептів *Triangle* та *Quadrangle*, в даному випадку не мають перетину, $|I^{\mathcal{I}}| = 0$, де $I = Triangle \sqcap Quadrangle$, їх оцінка відповідності за екземплярами також буде дорівнювати 0. В цьому випадку, напевне, більш достовірнішими будуть оцінювання ступенів відповідності концептів за їхніми властивостями або з використанням найменшого спільного покриття.

Слід зазначити, що наведений приклад грунтується на базовій дескриптивній логиці, що використовує лише конструктор перетину, а TBox фактично є таксономією. Тому, для його концептів LCS існує завжди і для будь-яких концептів C і D з цього TBox *LCS(C,D)=MSA(C,D)*. Зокрема, $Polygon$ =*LCS*($Triangle$, $Quadrangle$) = MSA($Triangle$, $Quadrangle$).

Функцію подібності концептів на базі LCS можна визначати на основі відстаней між концептами або перекриття розширень відповідних концептів (їхніх множин екземплярів).

$$dist(Triangle, Quadrangle) =_{def} |nf(Quadrangle)| + \\ + |nf(Triangle)| - 2 * \left|nf\left(lcs(Triangle, Quadrangle)\right)\right| = \\ |nf(Quadrangle)| + |nf(Triangle)| - \\ - 2 * |nf(Polygon)|=2+2-2*1=2$$

Використовуючи модель властивостей, оцінка подібності:

$$s(Triangle, Quadrangle) = \\ =_{def} \frac{2f(ftrs(Triangle) \cap ftrs(Quadrangle))}{f\left(ftrs(Triangle)\right) + f\left(ftrs(Quadrangle)\right)} = \\ = \frac{2 * 1}{3 + 3} = \frac{1}{3}$$

Зважаючи на те, що в даному випадку GCS=LCS=MSA:

$$\mathcal{S}(Triangle, Quadrangle) \\ = \frac{\min\left(|Triangle^{\mathcal{I}}|, |Quadrangle^{\mathcal{I}}|\right)}{|(LCS(Triangle, Quadrangle))^{\mathcal{I}}|} \\ * \left(1 - \frac{\left|\left(LCS(Triangle, Quadrangle)\right)^{\mathcal{I}}\right|}{|\Delta^{\mathcal{I}}|} * \right. \\ \left. * \left(1 - \frac{\min(|Triangle^{\mathcal{I}}|, |Quadrangle^{\mathcal{I}}|)}{\left|\left(LCS(Triangle, Quadrangle)\right)^{\mathcal{I}}\right|}\right)\right) \\ = \frac{\min(|Triangle^{\mathcal{I}}|, |Quadrangle^{\mathcal{I}}|)}{|Polygon^{\mathcal{I}}|} \\ * \left(1 - \frac{|Polygon^{\mathcal{I}}|}{|\Delta^{\mathcal{I}}|} * \left(1 - \frac{\min(|Triangle^{\mathcal{I}}|, |Quadrangle^{\mathcal{I}}|)}{|Polygon^{\mathcal{I}}|}\right)\right) = \\ = \frac{17}{47} * \left(1 - \frac{47}{49} * \left(1 - \frac{17}{47}\right)\right) = \frac{17}{47} * \frac{19}{49} \approx 0{,}14$$

Висновки

У роботі здійснено аналіз семантичних показників подібності, які класифіковані за підходами та моделями оцінювання. Наведені показники використовують семантичні висновки, такі, як, наприклад, перевірка екземплярів (що означає обчислення розширення концептів) заданого ABox. Внутрішня складність виразних мов ДЛ, таких, як ALC, обумовлює неефективність структурних підходів до аналізу. Тому визначення функцій подібності базуються на використанні теорії множин, що дозволяє застосовувати числові підходи, хоча й на символічному рівні представлення ДЛ.

Проаналізовані також моделі оцінювання та міри подібності на різних рівнях оцінювання. Основним є встановлення подібності між концептами (моделі понятійного рівня). Задачі обчислення ступеня подібності між екземплярами та між концептом та екземпляром зводяться до знаходження MSC та оцінювання подібності концептів.

Більшість наведених оцінок грунтуються на основі базових ДЛ, що підтримують лише констуктор перетину, але описані підходи можуть бути застосовані для будь-якої ДЛ. Це забезпечує базові сервіси міркувань, а саме: перевірку екземплярів та MSC (апроксимацію).

Представлені міри подібності можуть бути корисними при вирішенні багатьох задач різних типів, зокрема задач великих даних, таких як-от, пошук інформації в контексті термінологічних систем представлення знань, категоризація та класифікація даних тощо.

Література

1. Fellbaum, C. (Ed.). (1998). Wordnet: An Electronic Lexical Database. MA: MIT Press.
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook. Cambridge University Press (2003)
3. Staab, S., Studer, R., eds.: Handbook on Ontologies. International Handbooks on Information Systems. Springer (2004)
4. Thompson, K., Langley, P.: Concept formation in structured domains. In Fisher, D., Pazzani, M., Langley, P., eds.: Concept Formation: Knowledge and Experience in Unsupervised Learning. Morgan Kaufmann (1991)
5. Haussler, D.: Learning conjuntive concepts in structural domains. Machine Learning (1989) 7–40
6. F. Baader, R. K¨usters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, Proceedings of the 16th International Joint Conference on Artificial Intelligence, pages 96–101. Morgan Kaufmann, 1999.
7. F. Baader, R. Sertkaya, and Y. Turhan. Computing least common subsumers w.r.t. a background terminology. In V. Haarslev and R. M¨oller, editors, Proceedings of Proceedings of the 2004 International Workshop on Description Logics (DL2004). CEUR-WS. org, 2004.
8. R. Rada, H. Milli, E. Bicknell, M. Blettner, ”Development and Application of a metric on Semantic Nets”, IEEE Trans. on Systems, Man, and Cybernetics, 19(1): 17-30 (1989)
9. Mantay, T.: Commonality-based ABox retrieval. Technical Report FBI-HH-M-291/2000, Department of Computer Science, University of Hamburg, Germany (2000)
10. Collet, C., Huhns, M.N., Shen, W.M.: Resource integration using a large knowledge base in carnot. IEEE Computer 24 (1991) 55–62
11. Fankhauser, P., Neuhold, E.J.: Knowledge based integration of heterogeneous databases. In Hsiao, D.K., Neuhold, E.J., Sacks-Davis, R., eds.: Proceedings of the IFIP WG 2.6 Database Semantics Conference on Interoperable Database Systems (DS-5). IFIP Transactions, North-Holland (1992)
12. Bright, M.W., Hurson, A.R., Pakzad, S.H.: Automated resolution of semantic heterogeneity in multidatabases. ACM Transaction on Database Systems 19 (1994) 212–253
13. Tversky, A.: Features od similarity. Psycological Review 84 (1997) 327–352
14. Jang, J., Conrath, D.: Semantic symilarity based on corpus statistic and lexical taxonomy. In: Proceedings of the International Conference on Computational Linguistics. (1997)
15. Resnik, P.: Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. Journal of Artificial Intelligence Research 11 (1999) 95–130
16. Weinstein, P., Bimingham, P.: Comparing concepts in differentiated ontologies. In: Proceedings of 12th Workshop on Knowledge Acquisition, Modelling, and Management. (1999)
17. Rodr´ıguez, M.A., Egenhofer, M.J.: Determining semantic similarity among entity classes from different ontologies. IEEE Transaction on Knowledge and Data Engineering 15 (2003) 442–456
18. A. Tversky, “Features of Similarity”, Psychological Review 84(4): 327-352, 1977.
19. J. Lee, M. Kim, and Y. Lee. Information retrieval based on conceptual distance in is-a hierarchies. Journal of Documentation, 2(49):188–207, 1993.

20. D. Maynard, W. Peters, and Y. Li. Metrics for evaluation of ontology-based information extraction. In Proceeding of the EON 2006 Workshop, 2006.
21. P. Resnik, ”Using Information Content to Evaluate Semantic Similarity”, Proc. IJCAI 1995 : 448-453
22. G. Miller & W.G. Charles, ”Contextual correlates of semantic similarity”, Language and Cognitive Processes, 6, 1-28, 1991.
23. W. Cohen, A. Borgida, H. Hirsh: “Computing Least Common Subsumers in Description Logics”, AAAI 1992: 754-760
24. R. Kusters & R. Molitor, “Computing Least Common Subsumers in ALEN”, IJCAI 2001: 219-224
25. Claudia d’Amato, Steffen Staab, Nicola Fanizzi, F. Esposito: “Efficient Discovery of Services Specified in Description Logics Languages”, SMRR 2007
26. C. d’Amato, N. Fanizzi, and F. Esposito. A semantic similarity measure for expressive description logics. In A. Pettorossi, editor, Proceedings of Convegno Italiano di Logica Computazionale, CILC05, Rome, Italy, 2005
27. C. d’Amato, N. Fanizzi, and F. Esposito. A dissimilarity measure for ALC concept descriptions. In Proc. of the 21st Annual ACM Symposium of Applied Computing, SAC2006, 2006.
28. P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. Journal of Artificial Intelligence Research, 11:95–130, 1999.
29. A. Borgida, T. Walsh, and H. Hirsh. Towards measuring similarity in description logics. In I. Horrocks, U. Sattler, and F. Wolter, editors, Proceedings of the 2005 International Workshop on Description Logics (DL2005), volume 147 of CEURWorkshop Proceedings. CEUR-WS. org, 2005.

References

1. Fellbaum, C. (Ed.). (1998). Wordnet: An Electronic Lexical Database. MA: MIT Press.
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook. Cambridge University Press (2003)
3. Staab, S., Studer, R., eds.: Handbook on Ontologies. International Handbooks on Information Systems. Springer (2004)
4. Thompson, K., Langley, P.: Concept formation in structured domains. In Fisher, D., Pazzani, M., Langley, P., eds.: Concept Formation: Knowledge and Experience in Unsupervised Learning. Morgan Kaufmann (1991)
5. Haussler, D.: Learning conjuntive concepts in structural domains. Machine Learning (1989) 7–40
6. F. Baader, R. K¨usters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, Proceedings of the 16th International Joint Conference on Artificial Intelligence, pages 96–101. Morgan Kaufmann, 1999.
7. F. Baader, R. Sertkaya, and Y. Turhan. Computing least common subsumers w.r.t. a background terminology. In V. Haarslev and R. M¨oller, editors, Proceedings of Proceedings of the 2004 International Workshop on Description Logics (DL2004). CEUR-WS.org, 2004.
8. R. Rada, H. Milli, E. Bicknell, M. Blettner, ”Development and Application of a metric on Semantic Nets”, IEEE Trans. on Systems, Man, and Cybernetics, 19(1): 17-30 (1989)
9. Mantay, T.: Commonality-based ABox retrieval. Technical Report FBI-HH-M-291/2000, Department of Computer Science, University of Hamburg, Germany (2000)
10. Collet, C., Huhns, M.N., Shen, W.M.: Resource integration using a large knowledge base in carnot. IEEE Computer 24 (1991) 55–62
11. Fankhauser, P., Neuhold, E.J.: Knowledge based integration of heterogeneous databases. In Hsiao, D.K., Neuhold, E.J., Sacks-Davis, R., eds.: Proceedings of the IFIP WG 2.6 Database Semantics Conference on Interoperable Database Systems (DS-5). IFIP Transactions, North-Holland (1992)
12. Bright, M.W., Hurson, A.R., Pakzad, S.H.: Automated resolution of semantic heterogeneity in multidatabases. ACM Transaction on Database Systems 19 (1994) 212–253
13. Tversky, A.: Features od similarity. Psycological Review 84 (1997) 327–352
14. Jang, J., Conrath, D.: Semantic symilarity based on corpus statistic and lexical taxonomy. In: Proceedings of the International Conference on Computational Linguistics. (1997)
15. Resnik, P.: Semantic similarity in a taxonomy: An information-based measure and its ap-

plication to problems of ambiguity in natural language. Journal of Artificial Intelligence Research 11 (1999) 95–130
16. Weinstein, P., Bimingham, P.: Comparing concepts in differentiated ontologies. In: Proceedings of 12th Workshop on Knowledge Acquisition, Modelling, and Management. (1999)
17. Rodr´ıguez, M.A., Egenhofer, M.J.: Determining semantic similarity among entity classes from different ontologies. IEEE Transaction on Knowledge and Data Engineering 15 (2003) 442–456
18. A. Tversky, “Features of Similarity”, Psychological Review 84(4): 327-352, 1977.
19. J. Lee, M. Kim, and Y. Lee. Information retrieval based on conceptual distance in is-a hierarchies. Journal of Documentation, 2(49):188–207, 1993.
20. D. Maynard, W. Peters, and Y. Li. Metrics for evaluation of ontology-based information extraction. In Proceeding of the EON 2006 Workshop, 2006.
21. P. Resnik, ”Using Information Content to Evaluate Semantic Similarity”, Proc. IJCAI 1995 : 448-453
22. G. Miller & W.G. Charles, ”Contextual correlates of semantic similarity”, Language and Cognitive Processes, 6, 1-28, 1991.
23. W. Cohen, A. Borgida, H. Hirsh: “Computing Least Common Subsumers in Description Logics”, AAAI 1992: 754-760
24. R. Kusters & R. Molitor, “Computing Least Common Subsumers in ALEN”, IJCAI 2001: 219-224
25. Claudia d’Amato, Steffen Staab, Nicola Fanizzi, F. Esposito: “Efficient Discovery of Services Specified in Description Logics Languages”, SMRR 2007
26. C. d’Amato, N. Fanizzi, and F. Esposito. A semantic similarity measure for expressive description logics. In A. Pettorossi, editor, Proceedings of Convegno Italiano di Logica Computazionale, CILC05, Rome, Italy, 2005
27. C. d’Amato, N. Fanizzi, and F. Esposito. A dissimilarity measure for ALC concept descriptions. In Proc. of the 21st Annual ACM Symposium of Applied Computing, SAC2006, 2006.
28. P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. Journal of Artificial Intelligence Research, 11:95–130, 1999.
29. A. Borgida, T. Walsh, and H. Hirsh. Towards measuring similarity in description logics. In I. Horrocks, U. Sattler, and F. Wolter, editors, Proceedings of the 2005 International Workshop on Description Logics (DL2005), volume 147 of CEURWorkshop Proceedings. CEUR-WS. org, 2005.

Про автора:

Захарова Ольга Вікторівна,
кандидат технічних наук,
старший науковий співробітник.
Кількість наукових публікацій в українських виданнях – 31.
http://orcid.org/0000-0002-9579-2973.

Одержано: 02.04.2021

Місце роботи автора:

Інститут програмних систем НАН України,
проспект Академіка Глушкова, 40.
Тел.: 526 5139.
E-mail: ozakharova68@gmail.com.
Моб.тел.: +38(068)594756

УДК 004.75 https://doi.org/10.15407/pp2021.03.027

С.Я. Свистунов, П.І. Перконос, С.В. Субботін, Є.М. Твердохліб

НА ШЛЯХУ ДО СТВОРЕННЯ УКРАЇНСЬНОЇ НАЦІОНАЛЬНОЇ ХМАРИ ВІДКРИТОЇ НАУКИ

Наведені основні результати створення в Національній Академії наук України прототипу Національної хмари відкритої науки, інтегрованої до European Open Science Cloud.
Ключові слова: інформатизація наукових досліджень, хмарна інфраструктура, корпоративні інформаційні системи, European Open Science Cloud, національна хмара відкритої науки

Вступ

Сучасні наукові дослідження потребують величезних витрат та значних інформаційних ресурсів для реєстрації даних у процесі проведення експериментів, кількісного моделювання процесів, обробки накопичених даних. Тому найчастіше для виконання поточних наукових проектів установам не вистачає власних обчислювальних потужностей.

З іншого боку, специфіка процесу наукових досліджень та пошуковий характер наукової діяльності обумовлює нерівномірність завантаження обчислювальних ресурсів наукових установ і обчислювальних потужностей, які були задіяні для виконання проєкту, а після його закінчення простоюють. Проблему нестачі ресурсів для виконання проєкту можна вирішити за рахунок використання тимчасово вільних ресурсів в інших наукових установах. Однак пошук обчислювальних ресурсів дата центрів та встановлення договірних відносин є суттєвою організаційною проблемою, а використання таких ресурсів пов'язане з технічними проблемами розгортання та налаштування робочого середовища проєкту в дата-центрі.

Хмарні технології за рахунок віртуалізації забезпечують динамічний перерозподіл та ізоляцію фізичних обчислювальних ресурсів та використання їх як окремих одиниць з віддаленим доступом через Інтернет і таким чином спрощують технічні аспекти виділення вільних обчислювальних ресурсів дата-центрів в тимчасове користування іншим науковим установам, які їх потребують.

Організаційні проблеми пошуку вільних ресурсів серед наявних дата центрів та отримання їх в тимчасове користування можуть бути вирішені за рахунок об′єднання в єдину інфраструктуру, в якій відстежуються усі її наявні фізичні ресурси та їх завантаження, а також налагоджені механізми взаємовідносин між власниками дата-центрів та їхніми користувачами у відповідності з домовленостями на загальних принципах та згідно діючого законодавства.

Ще одним підходом до вирішення проблеми нестачі ресурсів для виконання проєкту є співпраця наукових колективів, працюючих над спільною науковою проблемою, а також повторне використання результатів досліджень навіть в інших галузях, які мають корисні методики та засоби досліджень, а також отримані результати експериментів та розрахунків. Це може значно скоротити обсяги робіт. Для цього результати наукових досліджень як у вигляді публікацій, так і первинних даних повинні бути доступні широкому колу науковців, тобто опубліковані в загальнодоступних репозиторіях.

1. Європейська хмара відкритої науки

Парадигма відкритих даних, відома як «будапештська ініціатива» [1], розвинута в декларації Європейської хмари відкритої науки (European Open Science Cloud - EOSC), яка поширює їх не тільки на публікації, а й на будь- які наукові дані та наукове співробітництво на базі використання хмарних технологій, а також передбачає практичні кроки розбудови спільної інфраструктури для колективного доступу до

інформаційних наукових ресурсів в Європі і в світі.

Відкрита наука є політичним пріоритетом Європейської комісії з 2016 року [2], і являє собою новий підхід до наукового процесу на основі нових способів поширення знань з використанням цифрових технологій і нових засобів для спільної роботи дослідників. Ідея парадигми «відкрита наука» фіксує системні зміни самого наукового-дослідного процесу - перехід від традиційних методів публікації результатів досліджень у наукових виданнях до обміну і використання результатів досліджень і даних, щойно вони стають доступними, з використанням цифрових технологій [3]. Для подальшої реалізації парадигми відкритої науки була створена консультативна група з представників дослідницького співтовариства, яка отримала назву - Європейська платформа політики відкритої науки (European Open Science Policy Platform) [4].

Для розвитку і поширення політики відкритої науки в Європі Європейська Комісія запропонувала створити Європейську хмару відкритої науки (European Open Science Cloud - EOSC). EOSC по суті є об'єднанням існуючих дослідницьких інфраструктур, репозиторіїв даних і пов'язаних з ними сервісів для підтримки наукових досліджень, роблячи дані досліджень сумісними і доступними відповідно принципами FAIR [5]. Ця мережа сховищ даних дозволяє дослідникам знаходити, використовувати і комбінувати пов'язані набори даних, забезпечуючи основу для створення нових інструментів з обробки даних, зокрема, на основі штучного інтелекту. Відкритість даних в EOSC дотримується принципу «якомога більш відкритими і за необхідності закритими». Це особливо важливо для наборів біомедичних, військових, конфіденційних, приватних і комерційних даних, які не можна відкрити негайно, повністю або взагалі будь-коли публікувати.

Розвиток EOSC за останні п'ять років можна розбити на кілька етапів: початковий етап, перехідний період та період формування структури EOSC асоціації.

Основне завдання початкового етапу створення EOSC (з 2017 по 2019 рік) - це формування архітектури, визначення основних вимог до сервісів і дослідницьких інфраструктур, які склали основу EOSC і формування команди виконавців, котрі беруть участь у проєкті. На цьому етапі Європейська комісія інвестувала близько 320 мільйонів євро для фінансування проектів в рамках програми Horizon 2020 для побудови основ EOSC. Ці інвестиції були спрямовані на розробку нового загальноєвропейського механізму доступу до існуючих електронних і дослідницьких інфраструктур, на координацію національних ініціатив і національних проєктів з метою підключення європейських дослідницьких інфраструктур до EOSC, на створення і впровадження принципів FAIR, а також для впровадження системи сертифікації FAIR даних. Для координації робіт були створені дві експертні групи, які представили рекомендації щодо архітектури EOSC [6], плану реалізації EOSC [7], а також рекомендації щодо впровадження принципів FAIR [8].

На початковому етапі було реалізовано понад 35 проєктів, які заклали основи архітектури побудови EOSC і продемонстрували різноманітність підходів і складність практичної реалізації. У проєкті EOSCpilot була запропонована структура і політика управління EOSC, а також розглянуті підходи до сумісності існуючих дослідницьких інфраструктур для різних наукових галузей [9]. Проект EOSC-hub об'єднав постачальників сервісів для створення єдиної точки входу для пошуку, доступу та використання широкого спектру обчислювальних і інформаційних ресурсів [10]. П'ять кластерних проєктів орієнтувалися на об'єднання дослідницьких інфраструктур в EOSC по різних наукових галузях, а саме: науки про навколишнє середовище ENVRI-FAIR [11], науки про життя EOSC-Life [12], астрономія і фізика елементарних частинок ESCAPE [13], дослідження фотонних і нейтронних пучків PaNOSC [14], соціальні та гуманітарні науки SSHOC [15]. П'ять регіональних проєктів були спрямовані на координацію зусиль національних проєктів із підтримки EOSC для європейських країн EOSC-Nordic [16], EOSC-

Pillar [17], EOSC-Synergy [18], ExPaNDS [19] і NI4OS-Europe [20]. Нарешті, в проекті HNSciCloud була створена гібридна хмарна платформа для підтримки високопродуктивних обчислень і обробки великих обсягів даних з використанням ресурсів комерційних провайдерів HNSciCloud [21], ARCHIVER [22] і OCRE [23].

Початковий етап створення EOSC був пов'язаний з програмою фінансування Horizon-2020, яка закінчилася в грудні 2020 року. 2018 році була опублікована дорожня карта реалізації EOSC, в якій представлена стратегія і шість основних напрямків реалізації: архітектура, дані, сервіси, політика доступу, інтерфейси, а також модель управління EOSC [24]. Ця дорожня карта не тільки охоплює перший етап реалізації EOSC (2018-2020 роки), але також визначала напрямки розвитку на другому етапі реалізації EOSC в рамках нової програми фінансування Horizon Europe на 2021-2027 роки. Дорожня карта передбачає створення пан'європейської федерації дослідницьких інфраструктур, об'єднаних навколо федеративного ядра, що забезпечує доступ до широкого спектру сервісів, які надаються на національному, регіональному та інституційному рівнях.

Із метою об'єднання розробників, провайдерів сервісів і користувачів та для забезпечення плавного переходу від першого до другого етапу впровадження EOSC була розроблена трирівнева структура управління, яка успішно працювала в 2019-2020 роках [25]. Основним елементом управління стала Виконавча рада EOSC (EOSC Executive Board), що складалася з восьми членів, які представляли найбільші дослідницькі інфраструктури, і трьох незалежних експертів. Основними завданнями Виконавчої ради було надання консультацій та підтримка стратегії розвитку, реалізації, моніторингу і звітності в EOSC [26]. Керуюча рада EOSC (EOSC Governance Board), що складалася з представників держав-членів ЄС та асоційованих членів ЄС, забезпечила контроль і підтримку діяльності Виконавчої ради [27]. Форум учасників проекту (EOSC Stakeholder Forum), що складався з представників організацій учасників, представників проєктів, забезпечував широке обговорення процесу реалізації EOSC [28]. Структура управління EOSC підтримувалася в проекті EOSCsecretariat.eu, який не тільки виконує функції секретаріату, а й забезпечує організацію заходів і збір пропозицій від спільноти щодо спільної розробки та реалізації EOSC [29].

У ході своєї роботи Виконавча рада визначила пріоритетні галузі наукових досліджень, які мають тематичні дослідні інфраструктури для включення їх в EOSC. Вона також створила шість робочих груп, що складаються з експертів учасників проектів, які виконувалися на початковому етапі реалізації EOSC, і представників організацій, зацікавлених у використанні сервісів EOSC [30]. Можна відзначити кілька, на наш погляд, провідних робочих груп. Група WG Architecture визначала технічні вимоги до дослідницьких інфраструктур, які необхідні для включення і спільного функціонування федерації EOSC, включаючи інтерфейси прикладного програмування (API), інфраструктуру автентифікації і авторизації (AAI) [31]. Група WG FAIR сконцентрувала зусилля на визначенні вимог для розробки, оцінки і сертифікації сервісів EOSC з метою забезпечення міждисциплінарної сумісності через FAIR [32]. Група WG Landscape проводила аналіз готовності наявних дослідницьких інфраструктур в Європі, які можуть бути підключені до EOSC [33].

Основні цілі Виконавчої ради та робочих груп були визначені в документі Стратегічний план реалізації (SIP) [34], а також у плані роботи на 2019-2020 роки [35]. Завершальним результатом роботи Виконавчого ради була розробка рекомендацій щодо механізмів та можливих форм управління на другому етапі реалізації EOSC в 2021-2027 роках.

На основі цих рекомендацій 29 липня 2020 року в Бельгії була створена EOSC Асоціація як некомерційна міжнародна асоціація (AISBL). 21 жовтня 2020 року було проведено EOSC Symposium, на якому були представлені проекти основних документів EOSC, зокрема, EOSC Association Statutes. 17 грудня 2020 року було проведено перше Засідання Генеральної Асамблеї

EOSC (EOSC General Assembly), на якому визначено її склад. EOSC Association наразі об'єднує 142 члени, з них: 21 учасник із дійсним мандатом від країни (Україна входить в це число) і 49 учасників у статусі спостерігача. Учасниками EOSC Association є провідні Європейські наукові організації, ресурсні центри, провайдери грід та хмарних сервісів, які беруть активну участь в реалізації Концепції Європейської хмари відкритої науки. Дійснім членом EOSC Association як офіційного представника від України є Інститут теоретичної фізики ім. М.М.Боголюбова НАН України [29].

Окремо слід відзначити результати, отримані в проєкті EOSC-hub, де представлено еталонну технічну архітектуру для EOSC, покликану полегшити доступ до сервісів, забезпечити технічні та організаційні умови для інтеграції і створення сервісів, а також для їх ефективного використання в наукових дослідженнях. Хоча формально основним завданням проекту EOSC-hub було проєктування, розробка і підтримка функціонування порталу EOSC [36], пілотна версія якого була введена в експлуатацію в листопаді 2018 року, в проєкті була забезпечена розробка та підтримка допоміжних компонентів порталу, таких як веб-сайт з його інформаційними розділами, інфраструктура автентифікації і авторизації (AAI), проксі-сервер, сумісний з архітектурою AARC Blueprint на основі технології EGI Check-in, єдиний каталог сервісів EOSC. На сьогодні каталог сервісів EOSC налічує близько 250 сервісів для дослідників, що працюють в різних галузях науки. У порталі EOSC-hub інтегровані додаткові інструменти для поліпшення взаємодії з користувачем і розширення його функціональних можливостей: служба підтримки користувачів, система моніторингу, а також система обліку використання ресурсів.

У проекті EOSC-hub були визначені принципи взаємодії та інтеграції нових сервісів і розроблені технічні вимоги, яким повинні задовольняти нові сервіси, які будуть інтегровані в портал. Ці вимоги ґрунтуються на інтерфейсах, які задокументовані і добре зарекомендували себе у використанні та інтеграції, та засновані на стандартах або API-інтерфейсах, що полегшують використання сервісів EOSC для спільнот користувачів.

Поряд із цим, у проєкті була проведена велика робота з інтеграції понад 30 тематичних сервісів великих дослідницьких спільнот як от: CLARIN, CMS/DODAS, ECAS/ENES, GEOSS, OPENCoastS, WeNMR, EO Pillar, DARIAH, LifeWatch. І тепер вони доступні європейським дослідникам через портал EOSC. В цілому ці тематичні сервіси успішно завершили 40 сценаріїв використання технічної інтеграції з 19 різними сервісами з портфеля послуг електронних інфраструктур EGI, EUDAT і INDIGO.

Крім того, в рамках проєкту було створено та забезпечено ефективну роботу 8 центрів компетенції для використання і спільної розробки сервісів для наступних дослідницьких спільнот: ELIXIR, Fusion (ITER), Argo, SeaDataNet, EISCAT_3D, EPOS-ORFEUS, LOFAR спільноти SKA, ICOS, eLTER і Disaster Mitigation.

Із впевненістю можна констатувати, що проект EOSC-hub представив прототип Європейського хмари відкритої науки і є хорошим прикладом для національних наукових співтовариств із розробки та впровадження національних стратегій хмари відкритої науки і дорожньої карти її реалізації.

2. Національна стратегія побудови хмари відкритої науки

Якщо притримуватися тверджень, що «EOSC - це середовище, яке має бути реалізоване для підтримки і забезпечення переходу до відкритої науки», то створення такого середовища не може бути виконано на порожньому місці, це має бути об'єднання вже наявних дослідницьких та е-інфраструктур країн - учасниць проєкту і забезпечена їхня спільна робота. Іншими словами, Європейська хмара відкритої науки - це не щось нове, що потрібно створити, а це спільне використання за новими правилами вже існуючих е- інфраструктур та дослідницьких інфраструктур .

Згідно проекту Концепції розвитку українських е-інфраструктур в Україні існують такі е-інфраструктури [37]:

· е-інфраструктури, які призначені для проведення розподілених обчислень, зо-

крема Національна цифрова інфраструктура, призначена для розподілених обчислень: Український Національний Грід (УНГ);

· е-інфраструктури, які призначені для забезпечення комунікації та мережевого зв'язку, зокрема Українська академічна мережа обміну даних (АМОД), Українська науково-освітня телекомунікаційна мережа (УРАН) та Українська академічна і дослідницька мережа ІФКС НАН України (УарНЕТ), призначені для забезпечення комунікації та мережевого зв'язку;

· е-інфраструктури, які призначені для накопичення, зберігання, систематизації та надання доступу до наукових даних, зокрема Національний репозиторій академічних текстів (НРАТ), призначений для накопичення, зберігання, систематизації та надання доступу до наукових даних.

Основним завданням е-інфраструктур є отримання, зберігання, обмін, управління та інтеграція наукових даних, їх глибинний аналіз, візуалізація, послуги обчислення та комунікації, а також інші послуги обробки інформації, які надаються через мережу Інтернет, не обмежуючись рамками однієї інституції.

Національна академія наук України має усі компоненти цієї архітектури, необхідні для формування єдиного відомчого інформаційного простору, який може стати основою Національної хмари відкритої науки [38], інтегрованого з EOSC.

Це і високошвидкісна Академічна Мережа Обміну Даних АМОД [39], яка може слугувати як комунікаційна основа, до якої приєднані майже всі установи НАН України, і має вихід в Європейську мережу GEANT.

Це грід-кластери НАН України [40], які можуть бути основою обчислювального ресурсу такої інфраструктури. З 2007 року по 2013 роки за цільової державної науково-технічної програми «Грід - інфраструктура та її використання», а з 2014 року по сьогодні за Програмою інформатизації Національної Академії наук України побудовано та функціонує Український національний Грід, який з 2012 року був інтегрований до Європейської Грід - інфраструктури на технічному рівні, а з 2020 року став асоційованим членом EGI.

Як показала практика формування EOSC [41], грід-кластери можуть бути природно адаптовані для використання як хмарного дата - центру шляхом встановлення відповідного проміжного програмного забезпечення [42]. Низка дата - центрів, а саме: Інститут теоретичної фізики ім.М.М.Боголюбова НАН України (ІТФ), Інститут програмних систем НАН України (ІПС), Інститут кібернетики імені В. М. Глушкова НАН України (ІК), Національний науковий центр Харківський фізико - технічний інститут НАН України (ХФТІ) вже мають досвід використання хмарного забезпечення інтероперабельного до Європейських інфраструктур. Окрім того дата - центри ІТФ, ХФТІ вже підключені до Європейської федеративної хмарної інфраструктури EGI [41], яка є складовою частиною EOSC.

В Україні підтримується Наукова електронна бібліотека періодичних видань НАН України (NASPLIB) [43], що об'єднує близько 500 видань. Вона є бібліотекою відкритого доступу і передбачає безкоштовний доступ читачів до наукової інформації в Інтернеті з правом проводити пошук, читати, завантажувати, копіювати, розповсюджувати, посилатися на повнотекстові статті, тощо. Тобто використовувати її законно, без фінансових, юридичних і технічних перешкод, що відповідає Будапештській Ініціативі Відкритого Доступу. До NASPLIB підключені міжнародні наукометричні та бібліометричні сайти, які надають статистичну інформацію з використання статей та журналів.

NASPLIB присутня у багатьох міжнародних реєстрах [44], зокрема: Registry of Open Access Repositories, OpenDOAR, Bielefeld University Library, інформаційний ресурс EOSC – OpenAire, Core та інших, що забезпечує взаємодію з ними та автоматичний глобальний доступ до публікацій, зареєстрованих в цих репозиторіях з використанням механізмів харвестінгу.

Однак, незважаючи на існування в НАН України системи колективного використання цінного наукового обладнання, дані, що отримуються в цих центрах, закриті і не мають цифрового формату, який дозволяє використовувати їх повторно.

Участь НАН України в процесі побудови Національної хмари відкритої науки передбачається Концепцією розвитку цифрової економіки та суспільства на 2018—2020 роки (схвалено Розпорядженням Кабінету Міністрів України від 17 січня 2018 р. № 67-р) та відповідає ключовим пріоритетам, що вказані в тексті Дорожньої карти інтеграції України до Європейського дослідницького простору (ERA-UA), яка була схвалена рішенням Колегії Міністерства освіти і науки України (протокол від 22.03.2018 № 3/1-7).

Основними напрямами гармонізації наукових ініціатив України з Європейським дослідницьким та інноваційним простором має бути:

· розбудова інтероперабельних цифрових інфраструктур для потреб закладів освіти та науки, підключення до освітньої мережі GEANT та системи розподілених обчислень, збору, зберігання та обробки даних європейської грід-інфраструктури;

· відкриття доступу до даних та публікацій, здійснених за рахунок державного фінансування.

Виходячи з вищезазначеного та з огляду на Європейський досвід побудови EOSC архітектурний устрій спільного інформаційного середовища для наукових досліджень (єдиний науковий інформаційний простір - ЄНІП) має враховувати наступні положення.

Інтеграційне середовище повинно мати комунікаційну основу, з допомогою якої здійснюється інформаційна взаємодія та переміщення даних у процесі опрацювання, з використанням узгоджених протоколів передачі даних.

Забезпечення гнучкого динамічного перерозподілу ресурсів для проведення наукових досліджень у разі необхідності потребує інтеграції технічних ресурсів окремих наукових установ та спільнот в єдиний пул – хмару загального користування.

Забезпечення накопичення та спільного використання даних передбачає розміщення та зберігання їх у загальнодоступних репозиторіях, що мають засоби зберігання, пошуку та отримання даних у всьому інформаційному просторі.

Забезпечення спільного використання напрацьованих методів та інструментів для обробки накопичених даних передбачає розміщення в єдиному просторі на FAIR-принципах сервісів, що втілюють ці принципи.

Враховуючи досвід побудови EOSC, в Україні розбудову Національної хмари відкритої науки (НХВН) доцільно розпочати, базуючись на існуючій грід-інфраструктурі, яка побудована за моделлю EGI. Першим етапом розбудови кореневої інфраструктури майбутньої НХВН може стати створення об'єднаної хмари УНГ, що сертифікована в об'єднаній хмарі EGI. Для формування об'єднаної хмари УНГ необхідно оновити комп'ютерне обладнання ресурсних грід-центрів, що сертифіковані в EGI для надання грід-сервісів.

Об'єднання Академічною мережею обміну даними (АМОД-УарНет) сертифікованих хмар, що функціонують на цих ресурсах з гарантованим пулом ресурсів загального користування, дозволить узгодити системи управління, стандарти та сервіси зі стандартами Європейської хмари відкритої науки, зокрема, проектом EOSC-Hub.

На другому етапі необхідно забезпечити інтероперабельність інфраструктури зі зрілими сервісами від цифрових бібліотек, архівів, медичних, екологічних та інших інформаційних систем і репозиторіїв, банків даних центрів колективного використання обладнання.

На шляху до створення національної хмари відкритої науки має відбутись розробка та впровадження порталу доступу до хмарним ресурсів і сервісів за аналогією з європейським порталом, розробленим в проекті EOSC-Hub.

3. Портал доступу до хмарної інфраструктури НАН України

Формування Національної хмарної інфраструктури пропонується виконувати шляхом добровільного надання ресурсів хмарними провайдерами Національної академії наук України. Станом на грудень 2020 року в Україні існує чотири інсталяції хмарних кластерів (рис. 1) під керуванням програмної системи OpenStack [42].

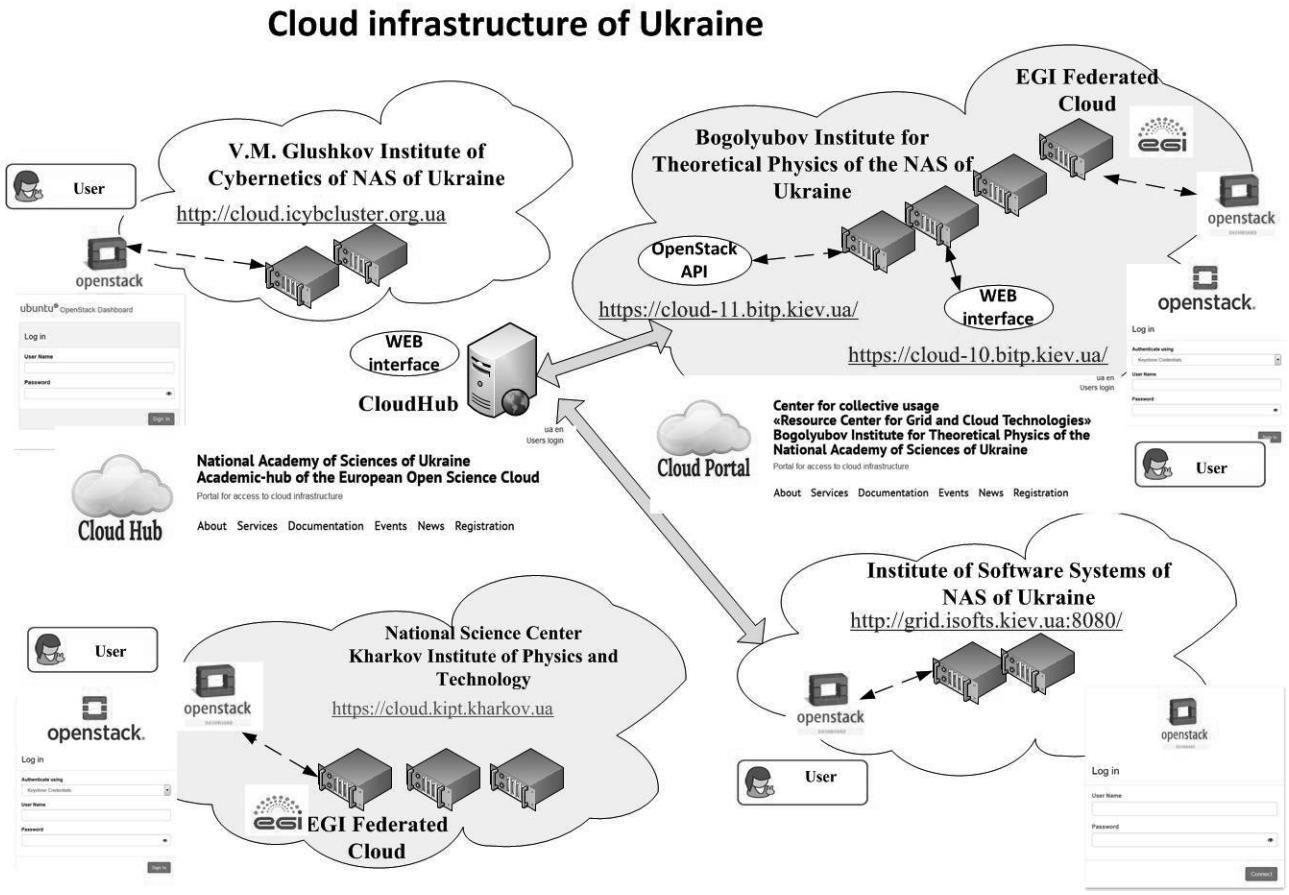


Рис.1. Хмарна інфраструктура НАН України

Два хмарних кластери ІТФ і ХФТІ, які сертифіковані і входять до складу EGI Federal Cloud, і хмарні кластери ІК і ІПС меншої обчислювальної потужності і обслуговують локальні проекти наукової спільноти інститутів.

Для тестування та практичної реалізації бізнес-процесів ресурсного забезпечення наукових проєктів два хмарних кластери ІТФ та ІПС були об'єднані порталом CloudHub для реалізації єдиної точки доступу до ресурсів. Обидва хмарні кластери були зареєстровані в порталі як складові хмарної інфраструктури із зазначенням виділених у розпорядження порталу обсягів ресурсів та адрес програмних сервісів їх хмарного програмного забезпечення.

Реєстрація хмарних ресурсних центрів виконується за межами порталу згідно бізнес-процесу, який складається з наступних кроків:

1. Адміністратор ресурсного центру надає заяву адміністратору порталу про реєстрацію дата центру за власною ініціативою або як відповідь на запит від адміністратора порталу.

2. Для реєстрації ресурсного центру в порталі адміністратор ресурсного центру надає наступні дані: назву ресурсного центру, параметри доступу, контактні дані адміністратора, обсяги ресурсів (віртуальні сервери, дискові накопичувачі тощо), які виділяються в загальне користування.

Отримання ресурсів для виконання проєкту здійснюється шляхом подання через портал заявок на ресурсне забезпечення, реєстрації проєкту і його користувачів у порталі. Для виконання проєкту виділяються наявні вільні ресурси дата центрів в рамках замовлених обсягів в заявці.

На даний час портал Cloud HUB забезпечує таку функціональність:

· підтримка реєстру дата центрів та облік виділених квот у спільне користування в дата центрах;

· підтримка реєстру наукових установ та користувачів і регламентований доступ до виконання процедур ресурсного забезпечення наукових проектів в ЄНІП;

· прийом від наукових установ заявок на відкриття проектів та узгодження обсягів ресурсів в ЄНІП для виконання проєкту;.

· прийом та виконання заявок на отримання ресурсів в ЄНІП на основі моніторингу квот та фактично задіяних ресурсів в дата центрах в активних наукових проєктах в реляційній базі даних порталу (рис.2).

· надання виконавцям через WEB інтерфейс інформації про виділені проєкту віртуальні машини та реквізити доступу до них.

У порталі CloudHub користувач отримує власний робочий кабінет (рис.3), який виконує функції робочого середовища та надає веб-інтерфейс для взаємодії з відповідним хмарним кластером щодо створення/управління/доступу до віртуальних серверів і створення необхідної інфраструктури проєкту, виходячи з доступних типових образів віртуальних машин, які завантажені в хмарному середовищі OpenStack. Крім того, портал дає можливість використовувати стандартний інтерфейс OpenStack - DashBoard для керування виділеними ресурсами.

Після виконання робіт віртуальна машина може бути зупинена, а її ресурси можуть бути використані на інші потреби. При цьому поточний стан віртуальної машини (snapshot) з усіма налаштуваннями може бути збережений на виділеному віртуальному носії в дата центрі.

Реалізовані в порталі CloudHub бізнес-процеси підтримки ресурсного забезпечення наукового проєкту пройшли тестування на прикладі проєкту розробки порталу та загального репозиторію програмного забезпечення.

Проєкту загального репозиторію були виділені ресурси в дата центрі ІПС, де була створена віртуальна машина на основі операційної системи Linux. До створеної машини забезпечено віддалений доступ, розгорнута система GitLab [45] для підтримки керування проєктами та контролю версій проєкту. Віртуальній машині виділено зовнішню IP-адресу. Запущена вона на постійній основі для забезпечення сервісу керування проєктами ЄНІП. У рамках проєкту були створені дві віртуальні машини – сервер порталу та сервер розробника. На сервері порталу розгорнуто WEB сервер APACH разом із СКБД mySQL. На сервері розробника розгорнуто інтегроване середовище WAMP для PHP платформи та система контролю версій GIT.

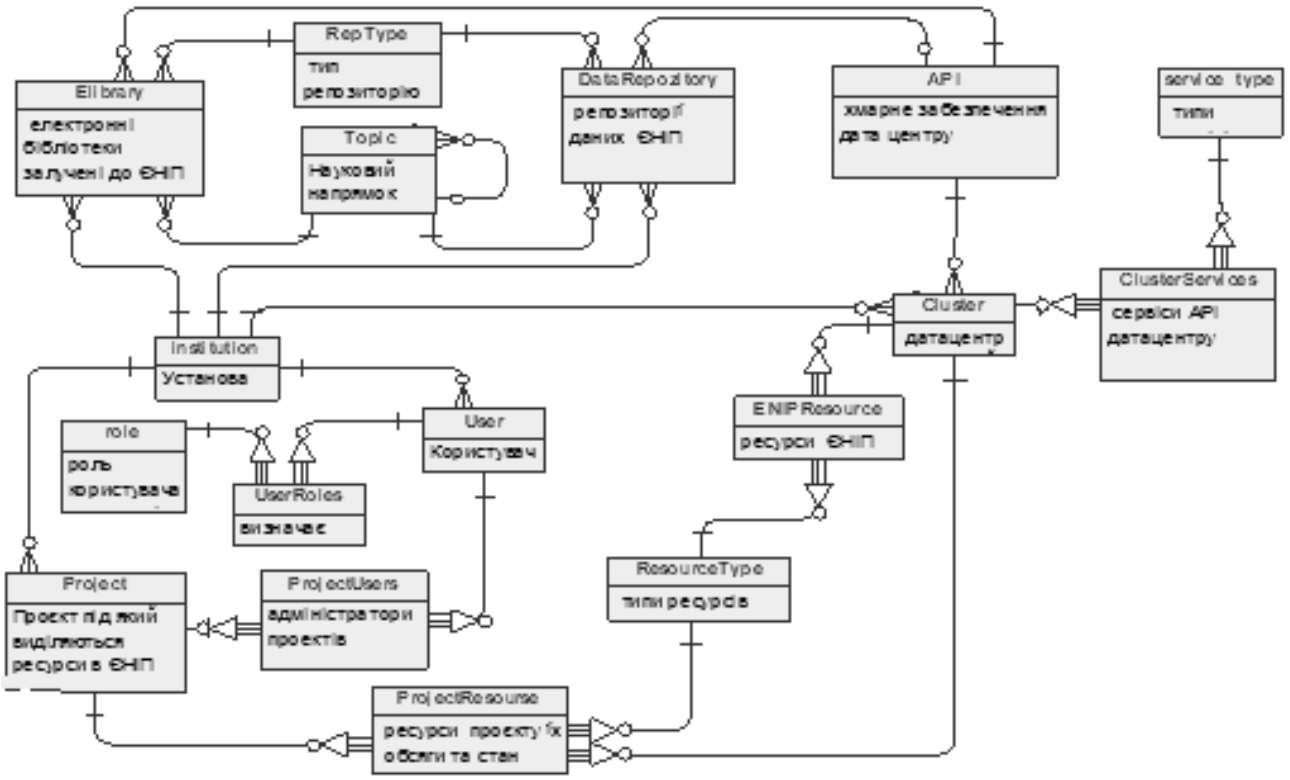


Рис.2. Реляційна структура даних обліку ресурсів ЄНІП

Рис. 3. Робочий кабінет користувача CloudHub

До серверів приєднуються блочні накопичувачі для розміщення поточних даних розробників та контенту порталу. Віртуальним серверам надано вихід в мережу Інтернет через локальну віртуальну мережу та маршрутизатор з однією зовнішньою IP адресою і налаштованими правилами маршрутизації портів на локальні адреси серверів. Розробники проєкту мають доступ до робочого оточення через віддалений доступ зі своїх робочих комп'ютерів.

Віртуальні сервери активуються за необхідності та призупиняються для вивільнення ресурсів хмари, якщо немає запитів користувачів.

Проєкт та його користувачі зареєстровані в репозиторії ПЗ ЄНІП, в якому організовано відстеження ходу проєкту та консолідація напрацювань окремих розробників засобами системи контролю версій.

Портал розроблено на базі системи керування контентом Cms sitograph [46], до якої додані прикладні компоненти, що реалізують облік ресурсів ЄНІП і взаємодію користувачів та адміністраторів дата центрів із хмарною інфраструктурою через Веб-інтерфейс. Прикладні компоненти реалізовані на платформі PHP із застосуванням бібліотеки взаємодії з сервісами openstack php-opencloud [47].

Висновки

У рамках Програми інформатизації НАН України започатковані роботи щодо створення спільного інформаційного середовища для наукових досліджень як прототипу Національної хмари відкритої науки, інтегрованої до Європейської хмари відкритої науки з огляду на Європейський досвід побудови EOSC.

Роботи базуються на основі власних досягнень, що відповідають вимогам часу, отриманих в результаті виконання проектів в рамках Програми інформатизації НАН України, Програми НАН України «Впровадження грід-технологій та побудова кластерів в НАН України» та Державної цільової науково-технічної програми «Впровадження і застосування грід-технологій на 2009-2013 роки», а саме:

· Академічна мережа обміну даних (АМОД) з великою пропускною спроможністю (10 ГБ), яка об'єднує всі наукові центри України, має вихід до потужних наукових мереж в Європі та забезпечує комунікаційне середовище для інформаційної підтримки діяльності НАН України. АМОД забезпечує близько 130 установ НАН України півтора десятком різноманітних послуг та сервісів, перелік яких постійно збільшується.

· Грід-інфраструктура, що розвиває цифрову науку, інтегровану в європейський простір. У провідних наукових установах НАН України функціонують 19 грід-вузлів та створено 3 потужні дата-центри, які забезпечують обчислювальні ресурси для обробки масштабних експериментів та розрахунків теоретичних моделей. Грід-інфраструктура забезпечує участь науков-

ців НАН України в міжнародних проектах (EGEE, Alice та ін.), на її основі розвиваються хмарні інфраструктури.

· Електронні бібліотеки та система інтеграції електронних бібліотек України, яка об'єднує 484 періодичних видань НАН України, 59 наукових видань, що виходять в електронному вигляді.

· Портал Cloud HUB, що забезпечує єдину точку доступу до ресурсів і сервісів хмарної інфраструктури НАН України.

Розбудова академічної хмарної інфраструктури дозволить поступово розширювати ступінь інтеграції до національного та міжнародного рівня, підвищуючи забезпеченість наукових досліджень ресурсами та ефективно їх використовувати.

Література

1. Budapest Open Access Initiative. https://www.budapestopenaccessinitiative.org/boai-10-translations/russian
2. Open innovation, open science, open to the world - a vision for Europe. https://ec.europa.eu/digital-single-market/en/news/open-innovation-open-science-open-world-vision-europe
3. D3.1: Policy Landscape Review. https://eoscpilot.eu/sites/default/files/eoscpilot-d3.1.pdf
4. Research and innovation. https://ec.europa.eu/research/openscience/index.cfm?pg=open-science-policy-platform
5. M. Wilkinson, M. Dumontier, I. Aalbersberg et al., The FAIR Guiding Principles for scientific data management and stewardship, Scientific Data volume 3, Article number: 160018 (2016)
6. Realising the European open science cloud. First report and recommendations of the Commission high level expert group on the European open science cloud. https://op.europa.eu/en/publication-detail/-/publication/2ec2eced-9ac5-11e6-868c-01aa75ed71a1
7. Prompting an EOSC in practice: Final report and recommendations of the Commission 2nd High Level Expert Group on the European Open Science Cloud (EOSC), 2018, doi:10.2777/112658 https://op.europa.eu/en/publication-detail/-/publication/5253a1af-ee10-11e8-b690-01aa75ed71a1
8. Turning FAIR into reality: Final Report and Action Plan from the European Commission Expert Group on FAIR Data, 2018, ISBN 978-92-79-96546-3, doi 10.2777/1524 KI-06-18-206-EN-N
9. EOSCPilot Project. https://eoscpilot.eu/
10. Services for the European Open Science Cloud. https://www.eosc-hub.eu/
11. Environment research infrastructures building FAIR services for research, innovation and society. https://envri.eu/home-envri-fair/
12. EOSC-life project. https://www.eosc-life.eu/
13. ESCAPE - The European Science Cluster of Astronomy & Particle Physics ESFRI Research Infrastructures. https://projectescape.eu/
14. The Photon and Neutron Open Science Cloud (PaNOSC). https://www.panosc.eu/
15. Social Sciences & Humanities Open Cloud (SSHOC). https://www.sshopencloud.eu/
16. EOSC-nordic project. https://www.eosc-nordic.eu/
17. EOSC-pillar project. https://www.eosc-pillar.eu/
18. EOSC-synergy project. https://www.eosc-synergy.eu/
19. ExPaNDS project. https://expands.eu/
20. National Initiatives for Open Science in Europe – NI4OS Europe. https://ni4os.eu/
21. The Helix Nebula Science Cloud (HNSciCloud) hybrid cloud platform. https://www.hnscicloud.eu/
22. Archiving and Preservation for Research Environments. https://www.archiver-project.eu/
23. Open Clouds for Research Environments. https://www.ocre-project.eu/
24. Staff Working Document - Guidelines on market analysis and the assessment of SMP under the EU regulatory framework for electronic communications networks and services. https://ec.europa.eu/research/ openscience/pdf/swd_2018_83_f1_staff_ working_paper_en.pdf
25. Setup and management of the EOSC Secretariat supporting the EOSC Governance. https://www.eoscsecretariat.eu/eosc-governance
26. The Executive Board of the EOSC. https://www.eoscsecretariat.eu/eosc-governance/eosc-executive-board
27. EOSC Governance Board. https://www.eosc-secretariat.eu/eosc-governance/eosc-governance-board

28. The EOSC Stakeholder Forum. https://www.eoscsecretariat.eu/eosc-governance/eosc-stakeholder-forum
29. The EOSC Secretariat. https://www.eoscsecretariat.eu/
30. EOSC Working Groups. https://www.eoscsecretariat.eu/eosc-working-groups
31. Architecture Working Group. Defining the technical framework required to enable and sustain an evolving EOSC federation of systems. https://www.eoscsecretariat.eu/working-groups/architecture-working-group
32. Working Groups. Implementing the FAIR data principles by defining the corresponding requirements for the development of EOSC services, in order to foster cross-disciplinary interoperability. https://www.eoscsecretariat.eu/working-groups/fair-working-group
33. Landscape Working Group. Mapping of the existing research infrastructures in Europe, which are candidates to be part of the EOSC federation. https://www.eoscsecretariat.eu/working-groups/landscape-working-group
34. European Open Science Cloud (EOSC) Strategic Implementation Plan, 2019, ISBN 978-92-76-09175-2, doi 10.2777/202370, KI-03-19-507-EN-N https://op.europa.eu/en/publication-detail/-/publication/78ae5276-ae8e-11e9-9d01-01aa75ed71a1
35. European Open Science Cloud (EOSC) work plan 2019-2020. https://op.europa.eu/en/publication-detail/-/publication/3c379ccc-ee2c-11e9-a32c-01aa75ed71a1
36. EOSC Portal - A gateway to information and resources in EOSC. https://eosc-portal.eu
37. Проект концепції розвитку українських дослідницьких інфраструктур, заснованих на технології комуникацій. https://mon.gov.ua/ua/news/stvoreno-proekt-koncepciyi-rozvitku-ukrayinskih-doslidnickih-infrastruktur-zasnovanih-na-tehnologiyi-komunikacij
38. КОНЦЕПЦІЯ формування Національної хмари відкритої науки, що інтегрована до Європейської хмари відкритої науки на 2018—2020 роки. Проект
39. Академічна мережа обміну даними. http://www1.nas.gov.ua/infrastructures/amod/Pages/about.aspx
40. Український національний грід. http://ung.in.ua/ua/
41. EGI: advanced computing for research. https://www.egi.eu/about/
42. Openstack. Documentation. https://docs.openstack.org/api-ref
43. Наукова електронна бібліотека періодичних видань НАН України. http://dspace.nbuv.gov.ua
44. Звіт за проектом Програми інформатизації НАН України за 2020 р. «Розвиток та супроводження Наукової електронної бібліотеки періодичних видань НАН України». http://programinform.nas.gov.ua/40
45. Система GitLab. https://gitlab.com/explore
46. Sitograph package. https://github.com/maxsv0/sitograph
47. Php-OpenCloud project. https://php-opencloud.readthedocs.io/en/latest/

References

1. Budapest Open Access Initiative. https://www.budapestopenaccessinitiative.org/boai-10-translations/russian
2. Open innovation, open science, open to the world - a vision for Europe. https://ec.europa.eu/digital-single-market/en/news/open-innovation-open-science-open-world-vision-europe
3. D3.1: Policy Landscape Review. https://eoscpilot.eu/sites/default/files/eoscpilot-d3.1.pdf
4. Research and innovation. https://ec.europa.eu/research/openscience/index.cfm?pg=open-science-policy-platform
5. M. Wilkinson, M. Dumontier, I. Aalbersberg et al., The FAIR Guiding Principles for scientific data management and stewardship, Scientific Data volume 3, Article number: 160018 (2016)
6. Realising the European open science cloud. First report and recommendations of the Commission high level expert group on the European open science cloud. https://op.europa.eu/en/publication-detail/-/publication/2ec2eced-9ac5-11e6-868c-01aa75ed71a1
7. Prompting an EOSC in practice: Final report and recommendations of the Commission 2nd High Level Expert Group on the European Open Science Cloud (EOSC), 2018, doi:10.2777/112658 https://op.europa.eu/en/publication-detail/-/publication/5253a1af-ee10-11e8-b690-01aa75ed71a1
8. Turning FAIR into reality: Final Report and Action Plan from the European Commission Expert Group on FAIR Data, 2018, ISBN 978-92-79-96546-3, doi 10.2777/1524 KI-06-18-206-EN-N

9. EOSCPilot Project. https://eoscpilot.eu/
10. Services for the European Open Science Cloud. https://www.eosc-hub.eu/
11. Environment research infrastructures building FAIR services for research, innovation and society. https://envri.eu/home-envri-fair/
12. EOSC-life project. https://www.eosc-life.eu/
13. ESCAPE - The European Science Cluster of Astronomy & Particle Physics ESFRI Research Infrastructures. https://projectescape.eu/
14. The Photon and Neutron Open Science Cloud (PaNOSC). https://www.panosc.eu/
15. Social Sciences & Humanities Open Cloud (SSHOC). https://www.sshopencloud.eu/
16. EOSC-nordic project. https://www.eosc-nordic.eu/
17. EOSC-pillar project. https://www.eosc-pillar.eu/
18. EOSC-synergy project. https://www.eosc-synergy.eu/
19. ExPaNDS project. https://expands.eu/
20. National Initiatives for Open Science in Europe – NI4OS Europe. https://ni4os.eu/
21. The Helix Nebula Science Cloud (HNSciCloud) hybrid cloud platform. https://www.hnscicloud.eu/
22. Archiving and Preservation for Research Environments. https://www.archiver-project.eu/
23. Open Clouds for Research Environments. https://www.ocre-project.eu/
24. Staff Working Document - Guidelines on market analysis and the assessment of SMP under the EU regulatory framework for electronic communications networks and services. https://ec.europa.eu/research/ openscience/ pdf/swd_2018_83_f1_staff_ working_paper_ en.pdf
25. Setup and management of the EOSC Secretariat supporting the EOSC Governance. https:// www.eoscsecretariat.eu/eosc-governance
26. The Executive Board of the EOSC. https:// www.eoscsecretariat.eu/eosc-governance/eosc-executive-board
27. EOSC Governance Board. https://www.eoscsecretariat.eu/eosc-governance/eosc-governance-board
28. The EOSC Stakeholder Forum. https://www.eoscsecretariat.eu/eosc-governance/eosc-stakeholder-forum
29. The EOSC Secretariat. https://www.eoscsecretariat.eu/
30. EOSC Working Groups. https://www.eoscsecretariat.eu/eosc-working-groups
31. Architecture Working Group. Defining the technical framework required to enable and sustain an evolving EOSC federation of systems. https://www.eoscsecretariat.eu/working-groups/architecture-working-group
32. Working Groups. Implementing the FAIR data principles by defining the corresponding requirements for the development of EOSC services, in order to foster cross-disciplinary interoperability. https://www.eoscsecretariat.eu/working-groups/fair-working-group
33. Landscape Working Group. Mapping of the existing research infrastructures in Europe, which are candidates to be part of the EOSC federation. https://www.eoscsecretariat.eu/working-groups/landscape-working-group
34. European Open Science Cloud (EOSC) Strategic Implementation Plan, 2019, ISBN 978-92-76-09175-2, doi 10.2777/202370, KI-03-19-507-EN-N https://op.europa.eu/en/publication-detail/-/publication/78ae5276-ae8e-11e9-9d01-01aa75ed71a1
35. European Open Science Cloud (EOSC) work plan 2019-2020. https://op.europa.eu/en/publication-detail/-/publication/3c379ccc-ee2c-11e9-a32c-01aa75ed71a1
36. EOSC Portal - A gateway to information and resources in EOSC. https://eosc-portal.eu
37. Draft concept for the development of Ukrainian research infrastructures based on communication technologies. https://mon.gov.ua/ua/news/stvoreno-proekt-koncepciyi-rozvitku-ukrayinskih-doslidnickih-infrastruktur-zasnovanih-na-tehnologiyi-komunikacij
38. THE CONCEPT of the formation of the National Cloud of Open Science, which is integrated into the European Cloud of Open Science for 2018-2020. Project
39. Academic data exchange network. http:// www1.nas.gov.ua/infrastructures/amod/Pages/about.aspx
40. Ukrainian national grid. http://ung.in.ua/ua/
41. EGI: advanced computing for research. https:// www.egi.eu/about/
42. Openstack. Documentation. https://docs.openstack.org/api-ref
43. Scientific electronic library of periodicals of the National Academy of Sciences of Ukraine. http://dspace.nbuv.gov.ua
44. Report on the draft Informatization Program of the National Academy of Sciences of Ukraine for 2020 “Development and maintenance of

the Scientific Electronic Library of Periodicals of the National Academy of Sciences of Ukraine". http://programinform.nas.gov.ua/40
45. Система GitLab. https://gitlab.com/explore
46. Sitograph package. https://github.com/maxsv0/sitograph
47. Php-OpenCloud project. https://php-opencloud.readthedocs.io/en/latest/

Одержано 22.04.2021

Про авторів:

Свистунов Сергій Якович,
кандидат технічних наук, завідувач відділу,
e-mail: svistunov@bitp.kiev.ua,
03186 , Київ, вул. Мартиросяна, буд.10/22, кв.45,
Кількість публікацій – 76,
ORCID ID 0000-0001-6502-4634

Перконос Петро Іванович,
науковий співробітник,
e-mail: Perkonos@nas.gov.ua,
04208, Київ, вул. В.Порика 7а, кв. 212,
Кількість публікацій – 25
ORCID ID 0000-0002-5958-0260

Субботін Сергій Васильович,
науковий співробітник,
e-mail: subbotin@protoka.kiev.ua,
03115, Київ, вул. Генерала Вітрука б. 3/11, кв. № 44,
Кількість публікацій – 15
ORCID ID 0000-0002-5958-0260

Твердохліб Євген Миколайович,
кандидат технічних наук, старший науковий співробітник,
E-mail: Eugine@nas.gov.ua,
03039, Київ, вул. Голосіївська б. 8, кв. № 53,
Кількість публікацій – 20.
ORCID orcid.org/0000-0001-6594-5468.

Місце роботи авторів:

Інститут теоретичної фізики
ім. М.М. Боголюбова НАН України,
03143, м. Київ вул. Метрологічна, 14-б
Тел.: +38(044) 521-34-94.
E-mail: svistunov@bitp.kiev.ua,

Інститут програмних систем НАН України
03680, Київ, проспект Академіка Глушкова, 40,
тел. +38(044) 526 6408
e-mail: Perkonos@nas.gov.ua

УДК 004.94 http://doi.org/10.15407/pp2021.03.040

В.А. Резніченко

60 РОКІВ БАЗАМ ДАНИХ (частина перша)

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 стандар¬тизація, управління транзакціями. Етап розширених реляційних баз даних присвячений опису темпоральних, просторових, дедуктивних, активних, об'єктних, розподілених та статистичних баз даних, баз даних масивів, машин баз даних і сховищ даних. На наступному етапі розкрита проблематика постреляційних баз даних, а саме, NOSQL-, NewSQL- і онтологічних баз даних. Шостий етап присвячений розкриттю причин виникнення, характерних властивостей, класифікації, принципів роботи, методів і технологій великих даних. Нарешті, в останньому розділі 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.

Етап 1. Становлення баз даних (1960 – 1970)

60-і роки – це період усвідомлення необхідності відокремлення даних від програм, кристалізації вимог до такої незалежної сукупності даних. Відтак, це період зародження й успішного становлення технологій баз даних, формування їхніх методологічних основ, становлення концепції моделі даних і появи перших двох класичних моделей – ієрархічної і мережевої, народження індустрії програмного забезпечення систем баз даних, а також – організаційного формування спільноти спеціалістів цієї галузі.

На початку 60-х років комп'ютери почали впроваджуватися на виробництві. Звісно, це здійснювали великі компанії, здатні придбати таке дороге обладнання. Комп'ютери почали використовуватись для автоматизації виробничих процесів, включно з обліком отримуваної сировини і деталей, виробленої продукції, персонала, тощо. Комп'ютери ставали інструментом збереження і обробки великих обсягів даних. До того ж стало очевидним, що технологія створення автоматизованих систем, за якої існував тісний звязок між даними і програмами, які їх використовують, є нежиттєздатною. Адже будь – які незначні зміни в структурі баз даних призводили до необхідності переписувати програми. З поступовим ускладненням структури даних і зростанням їхнього об'єму, збільшенням кількості користувачів, а також – інтенсивності використання даних, подібний підхід призводив до краху систем.

Це викликало усвідомлення того, що необхідно розірвати такий зв'язок і уможливити незалежне існування даних і програм, що й стало основою появи в інформатицінапрямку, який згодом отримав назву «бази даних».

Аби зрозуміти, в яких умовах зароджувалися бази даних, зазначимо, що це був час комп'ютерів фактично без операційної системи, з 64 КБ оперативної пам'яті. Носіями введення даних були перфокарти і перфострічки, а відповідальними за зовнішню память були переважно магнітні стрічки. І лише деякі компанії могли дозволити собі магнітні диски з об'ємом у 5 МБ і розмірами, що

перевищували тристулкову шафу разом із антресолями. Зрештою спілкування людини з комп'ютером відбувалося через пульт управління або, в кращому випадку, через друкарську машинку.

Завантаження жорсткого диску
на 5МБ компанії IBM

Системи IDS. 1960-го року невелика команда із General Electric, яка автоматизувала бізнес-процеси, розпочала проєктування системи Integrated Data Store (IDS) – інтегроване сховище даних під керівництвом Чарльза Бахмана (Charles William Bachman). Наприкінці 1962 року прототип цієї системи був завершений, а на початку 1964 року випущена перша промислова версія IDS [14-16]. Її поява знаменувала еру баз даних і зірковий шлях Бахмана.

Чарльз Бахман

В IDS було вперше втілено те, що нині вважається основними функціями системи управління базами даних. IDS виконувала функцію посередника між прикладними програмами і файлами, в яких зберігалися дані. Програми не могли напряму маніпулювати даними. Натомість вони мали звертатися до системи IDS, аби та виконувала відповідні дії від їх імені. Як і сучасні системи управління базами даних, IDS дозволяла створювати, зберігати і маніпулювати метаданими, хоча робилося це занадто примітивно. В IDS у найпростішому вигляді були реалізовані функціональні можливості, котрі згодом отримали назву «незалежності даних від програм». Бахман розробив у IDS інноваційну на той час систему «Диспетчер проблем» (Problem Controller), що стала прообразом системи управління транзакціями. В IDS також була спроєктована і реалізована система резервного копіювання і відтворення даних на магнітних стрічках. Зрештою, була передбачена функція заборони доступу до певних частин бази даних конкретним користувачам. IDS стала прообразом системи управління базами даних, що підтримувала мережеву модель даних. Система IDS розвивалась, удосконалювалась і використовувалась упродовж десятків років. І нині IDS використовується в деяких компаніях і демонструє відмінні результати продуктивності на терабайтних масивах даних.

1973 року Бахман був нагороджений найпрестижнішою в галузі інформатики премією Алана Тьюрінга за видатний внесок у технологію баз даних. Він був першим лауреатом премії Тьюрінга без ступеню доктора філософії, першим із досвідом роботи в галузі техніки, а не науки, й першим, чия кар'єра була повністю пов'язана з промисловістю, а не з наукою або науковим середовищем. Він також перший, хто отримав цю премію за роботу з базами даних.

Система IMS. 1965 року компанія IBM отримала замовлення на створення автоматизованої системи для обліку величезної кількості виробів, деталей і матеріалів, що мали використовуватися під час виконання космічної програми НАСА

«Аполон» - польоту людини на Місяць. Ця система попередньо отримала назву Information Control System – IMS (система управління інформацією). Відповідно до [1] в основу IMS було покладено модель даних, розроблену в середині 60-х років компанією North American Rockwell. 1968 року IMS була надана замовнику, а вже 1969 року стала доступною у сфері інформаційних технологій [17 - 19]. Відтоді й практично по сьогодні компанія IBM вдосконалює IMS, переносить на різні платформи і в різні операційні системи, розширює функціональні можливості. Це, власне, була перша успішна спроба створення промислового варіанту СУБД, хоча на той час так не називалася. Головним архітектором IMS був Верн Уоттс (Vern Watts). Він очолював цю роботу з моменту її проєктування аж до своєї смерті 2009 року.

Верн Уоттс

IMS підтримує ієрархічну модель даних. Вона складається зі схеми й екземплярів. На схемному рівні основним будівельним блоком є сегмент, який складається із сукупності полів. Сегменти зв'язуються направленими бінарними зв'язками. Сегмент, із якого виходить зв'язок називається батьківським, а який приєднується – дочірнім. Кожен сегмент може мати не більше одного батьківського сегменту й велику кількість дочірніх. Сегмент без батька називається кореневим, а без дочірніх сегментів – листям. На рівні екземплярів зв'язок між сегментами означає, що один екземпляр батьківського сегменту зв'язується з багатьма екземплярами дочірнього сегменту. Екземпляр ієрархічної структури містить один екземпляр кореневого сегменту. Отже, ієрархічна модель природно представляє зв'язки «один до багатьох». Слід зазначити, що жорстка формальна специфікація ієрархічної моделі даних відсутня, і вона, зазвичай висвітлюється так, як це було визначено в IMS.

Система Total. 1968 року Томас Нic (Tomas Nies), Клод Богардус (Cloud Bogardus) і Том Річлі (Tom Richley) заснували компанію Cincom Systems, а вже 1969-го було випущено першу версію СУБД Total [20].

Томас Нic

На думку багатьох користувачів і спеціалістів, система Total була серйозним конкурентом IMS на комп'ютерах IBM. На відміну від IMS і більшості інших СУБД того часу Total не обмежувалась одним типом комп'ютерів. Порівняно з IMS вправлятися з Total було доволі легко і ефективно.

Базовою структурою даних Total є дворівнева ієрархія, що містить один запис – власника (master) і велику кількість записів – членів (details). Ці типи записів можуть бути пов′язані так, щоби створювати складні структури даних. Ця структура нагадувала мережеву структуру перших версій IDS.

У Total підтримувалося узгодження із Cobol, Fortran, PL/1 і Assembler. Мова

маніпулювання нагадувала специфікацію Codasyl. Було реалізовано механізм захисту бази даних, який включав динамічну реєстрацію, періодичне резервне копіювання (дамп) і рестарт, запобігання одночасному оновленню даних. Підтримувався режим одночасної роботи багатьох прикладних програм. Було також реалізовано механізм незалежності на рівні окремих елементів даних. Для кожної програми можна було виділити доступну підчисельність бази даних з допомогою механізму, подібному підсхемам.

На початок 70-х років Total мала найбільшу кількість користувачів серед усіх діючих тоді СУБД. Вважається, що на початковому етапі компанія Cincom Systems зробила суттєвий внесок у розвиток СУБД.

Система Adabas. Adabas (database system – адаптивна система баз даних) – система управління базами даних компанії Software AG, Німеччина.

Уперше випущена для мейнфреймів IBM 1971 року. Початкова модель даних – на базі інвертованого індексу. Підхід Adabas відмінний від мережевої моделі даних, однак забезпечує можливість підтримки повної мережевої структури за рахунок неявних відносин. На момент створення мова маніпулювання Adabas являла собою розширення мов Cobol і PL/1. У 1980-і роки доповнена елементами реляційної моделі. На злеті популярності реляційних СУБД в середині 80-х років, вона була однією з найбільш затребуваних систем управління базами даних. IDS, IMS, Total, Adabas належать до складу так званих *навігаційних баз даних*. Цей термін був введений Чарльзом Бахманом у його статті [21], присвяченій отриманню премії Тьюринга. Суть цього класу полягає в тому, що записи даних можуть зв′язуватись між собою різними посиланнями, тим самим створюючи складну структуру даних, а мова маніпулювання дозволяє здійснювати довільну навігацію за цими посиланнями для отримання доступу до потрібних записів. Ідея навігаційних систем булла породжена появою магнітних дисків, які, на відміну від магнітних стрічок, перфострічок і перфокарт із послідовним доступом, надавали прямий доступ.

На завершення цього розділу зазначимо, що сам термін «база даних» (database) з′явився на початку 60-х років. На думку Вільяма Олле (T. William Olle) [1] цей термін уперше було введено у вжиток на симпозіумах, організованих компанією System Development Corporation (SDC) у 1963 і 1965 роках, хоча спочатку сприймався у доволі вузькому сенсі. В широкий ужиток термін увійшов лише на початку 1970-х років [22].

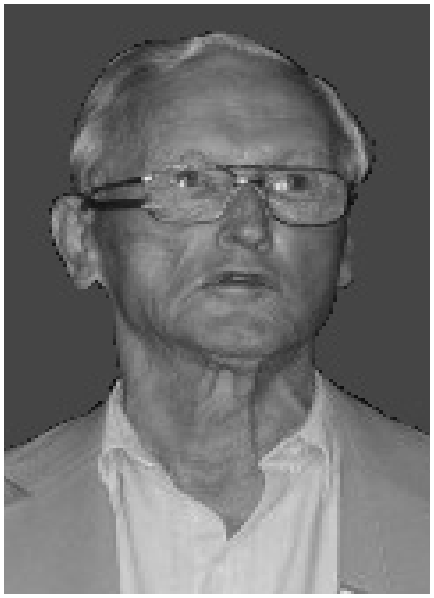
Етап 2. Бурхливий розвиток (1970 – 1980)

70-і роки – це роки бурхливого розвитку баз даних , створення основ технології баз даних. Вони ознаменувалися передовсім дослідженнями робочої групи CODASYL по базах даних (CODASYL DBTG), яка специфікувала мережеву модель, мови визначення і маніпулювання даними.В цей період було визначено і вивчено значну кількість моделей даних, включно із семантичними. 1876 року Петер Чен визначив ER – модель. Була специфікована трирівнева архітектура баз даних ANSI/X3/SPARC, яка стала класичною, здійснені дослідження щодо концептуального моделювання предметних галузей.Закладені основи індустріального виробництва СУБД та іншого програмного забезпечення баз даних. Зрештою, були реалізовані численні промислові СУБД, котрі виявилися затребуваними наступні кілька десятків років. 1973 року, як уже згадувалося вище, Чарльз Вільям Бахман був нагороджений найпрестижнішою в галузі інформатики премією Тьюрінга за видатний внесок у технологію баз даних.

До кінця 60-х років наукове співтовариство дійшло висновку, що системи управління базами даних (СУБД) стали центральною ланкою в автоматизованих інформаційних системах. Однак тоді ще не було повного усвідомлення того, що саме являє собою СУБД, які вимоги вони мусять задовольняти, які моделі даних мають підтримувати, яким архітектурним рішенням мають відповідати.

Але вже на початку 70-х років з′явилися перші звіти й статті, де робилися припущення із конкретних систем [23], а також формулювалися вимоги до СУБД [24,25].

Інфологічні й даталогічні моделі. Уже у 60-і роки вчені, котрі працювали в галузі інформаційних систем, зрозуміли, що в комп'ютерній системі мають бути представлені не лише дані, а і їхня семантика. В середині 60-х років шведський вчений Борже Лангефорс (Borje Langefors) ввів поняття інфологічної і дата логічної моделей (infological and datalogical models), які він розвивав упродовж 15 років [26 - 28].



Борже Лангефорс

Даталогічні моделі – це сукупність структурованих і взаємопов'язаних даних і способи оперування ними. Інфологічні моделі – це моделі представлення інформації (тобто – семантики) про дані. Ці терміни використовуються й по сьогодні, хоча з часом з'явився термін «семантична модель» як модель предметної галузі, призначена для представлення семантики предметної галузі на найвищому рівні абстракції. 1999 року Б. Лангефорс отримав престижну премію LEO за видатні досягнення в галузі інформаційних систем Міжнародної асоціації з інформаційних систем. А 2010 року шведська академія з інформаційних систем заснувала премію Б.Лангефорса за кращу докторську дисертацію Швеції в галузі інформатики й інформаційних систем.

Ідеї Лангефорса згодом були розвинуті й адаптовані до технологій баз даних шведським ученим Б. Сундгреном (Bo Sundgren) [29].

Бо Сундгрен

Архітектура баз даних ANSI/X3/SPARC. Із появою СУБД виникло нове поняття – схема даних (опис даних), яке відсутнє у файловій організації даних. Специфікація цієї схеми і маніпулювання даними виконуються вже мовними засобами СУБД – МОД (мова опису даних) і ММД (мова маніпулювання даними). Взаємодія СУБД із прикладною програмою здійснюється за допомогою розробки спеціального інтерфейсного модулю, в якому специфікуються об'єкти бази даних, потрібні цій програмі, а також необхідні операції над цими об'єктами. Зокрема, як це робиться в СУБД Adabas. Прикладна програма звертається до цього модулю через відповідну точку входу і передає йому певні параметри, що уточнюють запит. У відповідь програма отримує потрібні дані. Це так звана однорівнева архітектура. Цей єдиний рівень складає схема бази даних. Наступним кроком до вдосконалення було введення дворівневої архітектури. Суть її полягає в тому, що крім рівня схеми вводиться рівень підсхеми - фрагмента загальної схеми, який створюється для кожного додатку і описує дані, потрібні цьому додаткові. Дворівнева архітектура була прийнята в IMS. І нарешті в ANSI була визначена трирівнева архітектура баз даних, що стала класичною на багато десятиріч і про яку мова піде далі.

У листопаді 1972 року підкомітет SPARC (Standard Planning and Requirements

Committee) комітету X3 (Committee on Computers and Information Processing) Американського Національного Інституту Стандартів (ANSI) створив робочу групу ANSI/X3/SPARC DBMS для дослідження можливостей і розробки рекомендацій по стандартизації СУБД. Спочатку групу очолив Томас Стіл (Tomas B. Steel, Jr), а згодом Діонісіос Цикритзис (Dionysios Tsichritzis).

Діонісіос Цикритзис

Початковим завданням групи було дослідження питання, чи варто взагалі вирішувати проблему стандартизації СУБД. Якщо так, то що саме має бути стандартизоване. В результаті група дійшла висновку, що стандартизації можуть бути піддані лише інтерфейсні складові СУБД [30].

У зв'язку з цим було поставлено завдання визначення множини компонентів, з яких має складатися СУБД, інтерфейси, між якими могли б стати об'єктами стандартизації.В основу виявлення цих компонентів були покладені наступні концептуальні положення. По-перше, існує реальний світ, інформаційна модель якого має знайти своє відображення в базі даних. По-друге, враховуючи конкретні потреби, в свідомості людей відображаються їхні особисті уявлення про те, яким є реальний світ. Зрештою цей реальний світ матеріалізується у вигляді сукупності символів, у текстовому або електронному вигляді. Саме ця триєдність знайшла відображення в запропонованій групою зкомпонентної структури баз даних, що була названа трирівневою архітектурою баз даних ANSI-SPARC, і яка отримала загальне визнання серед розробників СУБД. Ця архітектура є актуальною й досі. Вона передбачає наявність концептуального, зовнішнього і внутрішнього рівнів. Концептуальний рівень призначений для опису концептуальної інформаційної моделі предметної галузі (ПГ). Зовнішній рівень визначає представлену користувачем БД. Це та частина БД, яка відповідає потребам конкретного користувача. Причому ця частина подається в зручному для користувача вигляді. Внутрішній рівень призначений для опису фізичного зберігання БД. Між цими рівнями існують відображення: концептуальний – зовнішній і концептуально – внутрішній. Ця трирівнева архітектура забезпечує необхідні умови для досягнення логічної і фізичної незалежності даних від програм. У свою чергу, дієвість механізмів опису відображень визначає ступінь достатності досягнення вищезгаданих двох видів незалежності. Результати діяльності цієї робочої групи були надані у звітах. 1977 року Томас Стіл отримав «Нагороду за видатні заслуги» (Distinguished Service Award) асоціації ACM.

Пропозиції КОДАСИЛ. Внесок CODASYL у технологію баз даних пов'язують із створенням мережевої моделі даних. 1967 року в КОДАСИЛ (CODASYL - Conference on Data Systems Languages) була створена спеціальна робоча група з питань баз даних (CODASYL Data Base Task Group — DBTG). Одним із першочергових завдань робочої групи було створення засобів управління базами даних для мови Кобол. Згодом ця задача була суттєво розширена і сформульована як розробка концепції, архітектури і мовних специфікацій баз даних загального призначення. 1971 року, усвідомлюючи важливість досліджень зі специфікації мовних засобів баз даних, було створено Комітет КОДАСИЛ із мови опису даних (CODASYL Data Description Language Committee). В результаті діяльності цих двох груп були опубліковані звіти [23, 33, 34], які викликали значний резонанс, були заслужено визнані фахівцями з баз даних і надовго стали зразком специфікації баз даних. У цих звітах, виходячи зі спільних

позицій і у тісному взаємозв'язку, вперше були строго специфіковані:

- мережева модель даних, ідеї якої були закладені Ч. Бахманом у системі IDS, що і отримала назву моделі даних КОДАСИЛ (CODASYL Data Model);
- трирівнева архітектура баз даних, що згодом була прийнята і розвинута в ANSI/X3/SPARC DBMS;
- мови опису даних (МОД) на всіх трьох рівнях (мова схеми, мова підсхеми, мова схеми зберігання);
- включені в МОД такі функції, як функція адміністрування, перевірки достовірності, управління доступом, налаштування, розподілу ресурсів, захисту даних, цілісності даних;
- відображення між схемою і підсхемою, а також схемою і схемою збереження;
- мова маніпулювання даними, призначена для навігації мережевною структурою з метою специфікації необхідного запису для його оновлення, видалення, або ж вставки нового запису.

За результатамироботи комітетів КОДАСИЛ були опубліковані численні матеріали, серед яких відмітимо монографію Вільяма Олле (T. William Olle) [1]. Підкреслимо, що пропозиції КОДАСИЛ були специфіковані для систем із включаючою мовою, тобто вони припускали, що робота з базою даних здійснювалась через мову програмування. Це повністю відповідало прийнятій тоді технології обробки даних і тому сприяло ефективній реалізації в існуючому вичислювальному середовищі.

Вільям Олле

Мережеві СУБД. Відповідно до специфікацій КОДАСИЛ була реалізована низка СУБД, серед яких: IDMS (Integrated Database Management System) компанії Cullinane Database Systems, що стала основною мережевою СУБД для мейнфреймів і найпопулярнішою в 70 – 80 –і роки минулого століття DMS 1100 (UNIVAC), IDS/II (Honeywell), DBMS 10/20 (DEC).

Концептуальне моделювання. У листопаді 1977 року комітет ISO з мов програмування прийняв рішення про створення робочої групи з питань дослідження різних аспектів використання концептуальних схем у системах управління базами даних з метою забезпечення основи для стандартизації в даній галузі. Спочатку цю групу очолив Т.Б.Стіл - молодший, а згодом Д.А.Жардін (D.A.Jardine). Результатом діяльності цієї групи став звіт, випущений 1982 року під редакцією Дж. Грийтусена (Joost J. Van Griethyusen).

Дж. Грийтусен

У звіті описуються роль і зміст концептуальної схеми, а також визначається зв′язок концептуальної схеми з інформаційним моделюванням і семантикою даних. Підкреслюється важливість точного визначення як статистичних, так і динамічних правил у концептуальній схемі:

- це спільна основа однозначного розуміння суті предметної галузі (ПГ) всіма зацікавленими сторонами;
- вона включає лише концептуально релевантні аспекти ПГ;
- це спосіб визначення допустимої еволюції інформаційної бази даних і

дозволеного маніпулювання інформацією про ПГ;

- це базис для інтерпретації зовнішніх і внутрішніх схем;
- це основа відображення зовнішніх схем у внутрішню і навпаки.

Моделі даних. Відповідно [13] термін «модель даних» почав використовуватися на початку 70-х років після публікації фундаментальної роботи Едгара Кодда (E. Codd) [59]. Однак ще в другій половині 60-х років почали з'являтися перші моделі даних. У результаті розвитку технології баз даних було запропоновано чимало засобів і методологій концептуального моделювання. Зокрем, серед них наступні моделі: модель «Об'єктів – ролей» (ОРМ – Object – Role Model) Екхарда Д.Фолкенберга (Folkenberg, Eckhard D.) [36, 37], яка була розвинута іншими вченими (С. Нейссен, Р.Меерсман, Д.Вермейр, Т.Халпін (Sjir Nijssen, Robert Meersman, Dirk Vermeir, Terry Halpin). ORM передбачає представлення інформаційної моделі у вигляді об'єктів (сутностей), котрі відіграють ту чи іншу роль (представлені у вигляді зв'язків між об'єктами). На відміну від об'єктно-орієнтованого підходу і підходу сутність-зв'язок ORM не передбачає існування атрибутів, вони подаються у вигляді ролей фактів, які разом із правилами моделюються у вигляді природних пропозицій, легких для розуміння і перевірки користувачами.

Екхард Д.Фолкенберг

Модель даних, заснована на бінарних зв'язках. Біля витоків походження моделі бінарних зв'язків (BR - Binary Relations) були праці таких авторів, як Абріаль [38] (семантична бінарна модель), Браччі [39], Дурхольц [40]. Суть цього підходу до моделювання в тому, що будь-який «елемент» інформації представляється з допомогою екземплярів бінарних асоціацій, тобто висловлювань, до складу яких входять тільки два терми. Зокрема, М.Сенко в межах проекту DIAM (Data Independence Access Method) визначив бінарну мережеву модель, розробив на базі цієї моделі мову FORAL і дослідив можливості користувальницького інтерфейсу, який на ній базується [41, 42, 43].

Семантичні моделі. Відзначимо роботи Дж. Сміта і Д. Сміта по моделях абстракції, агрегації і узагальнення даних [44, 45], а також семантичну модель даних SDM Хамера і МакЛеода [46]. У статті [47] наводиться перелік близько 20 семантичних моделей баз даних. Результатом розвитку моделей даних до початку 80-х років наведені в широко відомій монографії Д.Цикритзиса і Фреда Лоховскі (F.Lochovsky) [2].

Фред Лоховскі

ER – модель. Водночас, найбільшу популярність заслужено здобув підхід сутність – атрибут - зв'язок, названий як підхід сутність - зв'язок (ER – підхід). Свій початок він бере від діаграм структур даних Бахмана [48], а також моделі Інглеса [49].

Вперше найширше цю модель описав П.П.Чен (Peter Pin- Shen Chen) [50].

Петер Чен

ER – модель даних стала загальновизнаною в світі і є основою багатьох методик системного аналізу, концептуального моделювання й проєктування баз даних. Вона базується на простій ідеї, що структурна складова концептуальної моделі предметної галузі може бути представлена у вигляді сутностей, атрибутів і зв′язків. Сутність – це будь-який реальний або абстрактний об′єкт довільної природи, який представляє самостійний інтерес. Атрибут – це властивість сутності, що сприяє якісному або кількісному її опису, ідентифікації, класифікації або відображенню її стану. Нарешті зв′язок – це певна асоціація між різними сутностями (класами сутностей), що становить певний інтерес.

Після публікації статті Чена з′явилося чимало статей, присвячених дослідженню різних аспектів ER - моделювання предметних галузей. Наприклад, в загальному випадку припускається існування n – арних зв′язків, а Річард Баркер (Barker Richard) запропонував ER модель тільки з бінарними зв′язками [51], яка має певні переваги.

У зв′язку з широким використанням ER- моделі [3] було запропоновано багато різних її розширень і узагальнень [52 - 55], які зрештою привели до визначення ієрархічної ER- моделі (ER- моделі вищого порядку) [55]. У статті [56] ER-модель розширена елементами семантизації даних. Також була запропонована темпорально-розширена ER- модель [57], яка дає можливість включати темпоральну інформацію в концептуальну інформаційну модель і представляти її в реляційній моделі. Для підтримки темпоральних запитів мова SQL була розширена можливостями визначення, пошуку й управління історичними відносинами. Із часом було запроваджено ще кілька темпоральних er – моделей, огляд яких наведено в [58]. Зрештою, існує просторова ER - модель (див.: «Просторові бази даних»).

Етап 3. Епоха реляційних баз даних (1970 - 1990+)

На початку 80-х років з'явилися перші промислові реляційні СУБД, які до кінця 80-х стрімко завоювали ринок і стали панівними практично на всіх поширених апаратно-програмних платформах і не втратили свою перевагу й по сьогодні. Попри це, основи реляційної моделі даних і реляційних СУБД були закладені в попередньому десятиріччі, родоначальником їх був Едгар Франк Кодд. Він визначив реляційну структуру даних, алгебру і обчислення, заклав основи теорії залежностей і нормальних форм, сформулював вимоги реляційності баз даних. Ці та інші дослідження кінець кінцем привели до створення теорії реляційних баз даних. Бази даних перетворилися з описової науки у формальну.

1981 року Едгара Франка Кодда було нагороджено премією Тьюрінга за фундаментальний і тривалий внесок в теорію і практику систем управління базами даних, особливо реляційного типу. Було відкрито багато проєктів із дослідження і створення експериментальних СУБД, запропоновано велику кількість мов запитів реляційних баз даних, вивчено питання оптимізації виконання запитів, структури зберігання, методів доступу, захисту, збереження цілісності.

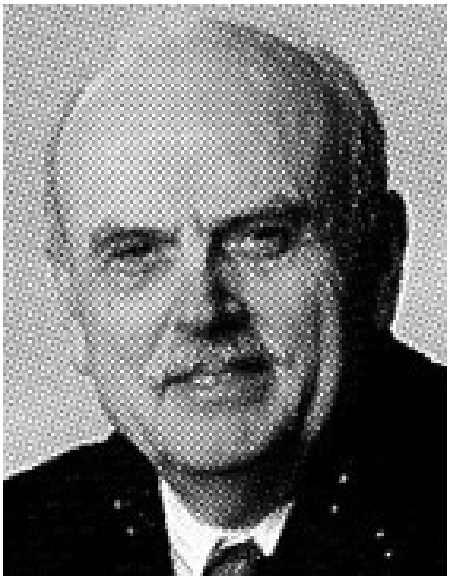
1986 року з'явився перший стандарт SQL і відтоді він став єдиною офіційною мовою зовнішнього інтерфейсу реляційних СУБД.

Було проведено численні дослідження з управління транзакціями, за які

1998 року Джеймс Ніколас Грей отримав премію Тьюрінга.

Реляційні бази даних. У 70-х роках учені, які працювали у сфері баз даних, були переконані, що майбутнє баз даних – за створенням усе складніших структур даних, які дозволяли б адекватно представляти інформаційну модель даних довільних предметни х галузей. Висловлювалися думки, що найближчим часом структури будуть настільки складними, що в базах даних співвідношення корисної інформації та тієї, що її підтримує, буде 1 : 30.

Внесок Е.Ф.Кодда в реляційні бази даних. І ось на цьому тлі 1970 року публікується стаття [59] маловідомого на той час британського вченого Едгара Франка Кодда (Edgar Frank Kodd) з компанії IBM, в якій він запропонував простішу структуру даних. Ця структура являла собою одномірну, плоску, нормалізовану таблицю.



Едгар Франк Кодд

Одномірність означає, що існує лише одна, горизонтальна шапка і не може бути вертикальної, як, скажімо, в навчальних планах ВУЗу. Плоскість свідчить про те, що в шапці не має бути полів, що складаються з багатьох підполів. Приміром, аби поле ПІБ складалося з підполів Прізвище, Ім′я, по-Батькові. І, нарешті, нормалізованість свідчить про те, що в осередках таблиці може бути тільки атомарне (єдине) значення. Така структура дістала назву реляційного відношення, бо вона нагадує математичне поняття відношення. Також було домовлено вважати, що такі відносини існують у першій нормальній формі (First Normal Form – 1NF). У цій же праці він обґрунтував існування двох сімейств реляційних мов, які пізніше були названі реляційним обчисленням і реляційною алгеброю.

1871 року Кодд публікує статтю [60], в якій наводить приклад того, як логіка обчислення предикатів може бути використана для створення високорівневої мови реляційної бази даних.

Описана ним мова ALPHA була першою мовою класу реляційного обчислення. Хоча ALPHA не була реалізована, однак вона мала значний вплив на створення наступних комерційних реляційних мов.

1972 року Кодд публікує наступну свою важливу статтю [61], де він:

- дає формальне визначення реляційної алгебри і реляційного обчислення (кортежно - орієнтованого);
- формулює тезу реляційної повноти селективних можливостей мов запитів до реляційної бази даних на основі реляційного обчислення. Ця теза була одностайно сприйнята вченими світу і в подальшому всі створювані мови запитів перевірялися на реляційну повноту;
- наводить алгоритм редукції довільного вираження реляційного обчислення в семантично еквівалентне вираження реляційної алгебри, тим самим встановлюючи її реляційну повноту. Цей результат згодом було названо теорією Кодда. Пізніше – Палермо (Palermo) [62] удосконалив цей алгоритм з точки зору підвищення його ефективності.

Реляційна модель відпочатку критикувалася за простоту її структури. Це, зокрема, відбулося й на конференції 1974 року «SIGMOD Workshop on Data Description, Access and Control», де виникли дебати між прихильниками реляційного і мережевого підходів, головними спікерами яких виступили Кодд і Бахман. Позиція Кодда на цих дебатах відображена у статті [63]. Зрештою, реляційна модель здобула загальне визнання. Це можна пояснити тим, що в ній вдалося сформулювати мови високого рівня (алгебра,

обчислення). А це дозволило найбільш повно вирішити ту основну проблему, яка була поставлена перед базами даних, а саме – досягнення незалежності даних від програм. У свою чергу, підвищення складності структури даних призводить до неминучого зниження рівня мови маніпулювання, що знижує можливості досягнення такої незалежності.

Намагаючись надати додаткові можливості, у праці [64] Е. Кодд запропонував підвищити семантику реляційної моделі, ідеї якої використовуються й нині в комерційних реляційних СУБД.

Теорія залежності і нормальних форм. Реляційна модель дала серйозний поштовх для розвитку проєктування баз даних. Уперше задача логічного проєктування БД дістала строго формальний підхід. Суть цієї теорії полягала в тому, щоб на основі аналізу різних видів залежностей (обмежень цілісності), які існують всередині реляційних відносин і між ними, виявляти небажані ситуації та усувати їх за допомогою обгрунтовани х процедур еквівалентних перетворень. Зазвичай такою процедурою є декомпозиція відносин, тобто розмежування відносин на декілька. Основоположником цієї теорії став Е.Ф.Кодд, опублікувавши праці [65 - 67]. В них він визначив поняття функціональної залежності (Functional Dependency – FD) в реляційному відношенні, сформулював так звані аномалії маніпулювання відношеннями, виявив два небажані різновиди FD і транзитивні FD, котрі породжують ці аномалії. А саме – неповні FDi транзитивні FD, і запропонував процедуру декомпозиції, що усуває ці різновиди FD у результуючих відношеннях. Відношення, де відсутні неповні FD, отримали назву відношень у другій нормальній формі (2NF), а там, де відсутні неповні й транзитивні FD – у третій нормальній формі (3NF).

1981 року Кодд був нагороджений премією Тьюрінга за фундаментальний і тривалий внесок в теорію і практику систем управління базами даних, особливо реляційного типу. Кристофер Дейт написав книгу[68] – історичний огляд наукового внеску Кодда в реляційну технологію.

З точки зору структури функціональних залежностей 3NF все ж мала певні аномалії. З огляду на це, 1974 року Кодд разом із Раймондом Бойсом (Raymond F. Boyce) запропонували підсилити 3NF. Результована нормальна форма дістала назву нормальної форми Бойса – Кодда (Boyce – Kodd Normal Form – BCNF) [69].

Раймонд Бойс

Як слушно зазначав Дейт, спочатку цю нормальну форму визначив Ян Хіт (Ian Heath) у статті [70]. Також зазначимо, що в цій статті він довів теорему про декомпозицію без втрат реляційного відношення за наявності FD, тобто декомпозиції, яка є еквівалентною за даними. Цю теорему було названо його ім′ям (Теорема Хіта). Вона застосовується при проведенні відношень в 2NF, 3NF і BCNF.

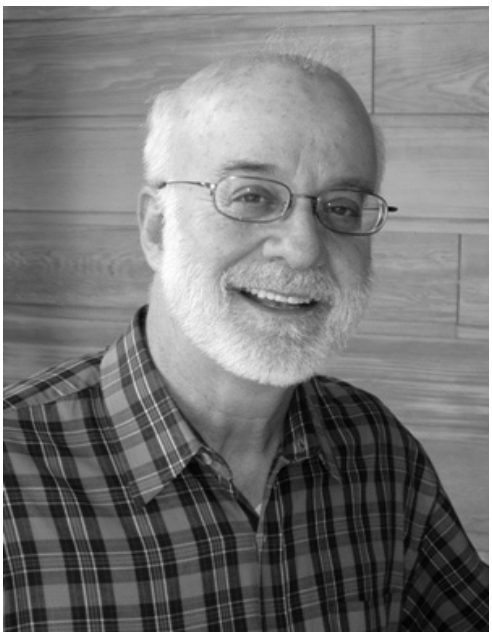
Ян Хіт

1974 року Вільям Армстронг (William Ward Armstrong) у статті [71] запропонував систему аксіом FD (мінімально повний набір правил виводу нових FD із заданих). Вони дістали назву аксіом Армстронга. Ці аксіоми дозволили визначити і дослідити такі поняття з системи FD, як виводимість, повнота, замикання, (мінімальне) покриття, еквівалентність. Отримані в цьому напрямку результати сприяли вирішенню задачі автоматизації проєктування баз даних.

Вільям Армстронг

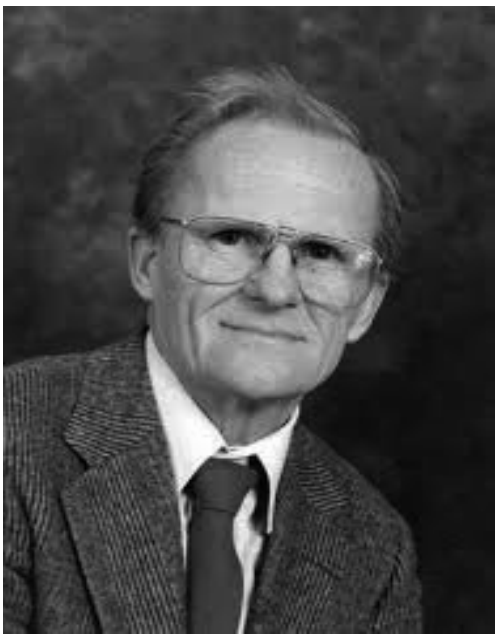
1977 року Рональд Феджин (Ronald Fagin) у статті [72] визначив новий вид залежності – багатозначну залежність (multivalued dependency MVD), наявність якої у відношеннях викликає аномалії маніпулювання. Запропонована ним форма, що усуває цю ситуацію, була названа четвертною нормальною формою (Forth Normal Form – 4NF), а алгоритм приведення в 4NF базувався на доведеній ним теоремі (теорема Феджина). В наступній статті [73] була запропонована повна система аксіом MVD, а також дві аксіоми, які пов′язують FD і MVD (виведення MVD із FD і навпаки).

Зазначимо, що окрім Феджина багатозначну залежність досліджував також Заніоло [74]. Крім того, Делобел [75] визначив поняття «ієрархічної декомпозиції першого порядку», яке також пов′язане з концепцією багатозначної залежності.



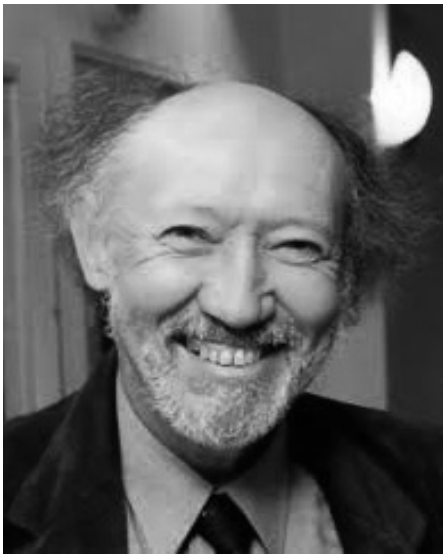
Рональд Феджин

1978 року Йорма Ріссанен (Jorma Rissanen) визначив залежність за з′єднанням (join dependency – JD) [76], яка стала узагальненням MVD (MVD є бінарною JD). На її основі Феджин у статті [77] визначив і дослідив проєкційно-з′єднувальну нормальну форму (Projection-Join Normal Form – PJ/NF), яка з часом дістала назву п′ятої нормальної форми (Fifth Normal Form – 5NF).



Йорма Ріссанен

Зрештою, Кристофер Дейт (Christopher Date) визначив шосту нормальну форму (Sixth Normal Form – 6NF) як форму, де відсутні нетривіальні залежності із з′єднання. Як підкреслюють численні дослідники, ця нормальна форма виявилася корисною в темпоральних базах даних. За ствердженням Дейта [78], 6NF рівносильна доменно-ключовій нормальній формі (DK/NF) Феджина (див. далі).



Кристофер Дейт

Наведені вище залежності і нормальні форми належать до так званих класичних. Наведемо ще кілька визначених і досліджених видів залежностей. Із їх докладним аналізом та структурою взаємозв′язків між ними можна ознайомитися в роботі [79]:

- поліпшена 3NF (Improved 3NF) [80];
- нормальна форма елементарного ключа (Elementary Key Normal Form – EKNF) [81];
- нормальна форма суперключа (Super Key Normal Form – SKNF) [82];
- приведена форма 5NF (reduced 5NF – 5NFR) [83];
- нормальна форма без надлишковості (Redundancy Free Normal Form – RFNF) [84];
- нормальна форма із суттєвими кортежами (Essential Tuple Norm Form – ETNF) [85];
- доменно-ключова нормальна форма Феджина (Domain – Key Normal Form – DK/NF) [86];
- ієрархічна залежність [87] та її зв′язок із ієрархічною структурою даних [88];
- залежність по включенню і нормальні форми по включенню (Inclusion Normal Forms) [90 - 92].

Насамкінець зазначимо, що ми навели лише незначну кількість досліджених залежностей. У книзі [93] наведений перелік понад 600 статей, присвячених теорії залежностей і нормальних форм, а в монографії [94] аналізується близько 90 залежностей.

Мова запитів реляційної моделі. Реляційна модель дала суттєвий поштовх дослідженням зі створення мов запитів. У своєму огляді [95] Дональд Чемберлен (Donald D. Chemberlin) запропонував наступну класифікацію мов реляційних баз даних: мови реляційної алгебри, мови реляційних обчислень, графічні мови і мови, орієнтовані на відображення. Даємо короткий огляд мов цих класів.

Дональд Чемберлен

Мови реляційної алгебри. Були запропоновані й експериментально опробовані наступні мови/системи, що базуються на реляційній алгебрі: система VACAIMS [96] розроблена в MIT, системи IS/1 PRTV (Peterlee Relational Test Vehicle) [98], розроблені в науковому центрі IBM в Пітерлі (Англія), система RDMS [99] створена в дослідницькій лабораторії General Motors. В багатьох системах розширюється набір операцій реляційної

алгебри шляхом введення специфічних. Паралельно проводились дослідження з оптимізації виконання виражень реляційної алгебри [101 - 104]. В [105] наводиться широкий огляд досліджень з аналізу складності операцій та оптимізації записів у реляційних базах даних.

Мови реляційного обчислення. Як ми вже зазначали вище, першою мовою запитів реляційної моделі була мова ALPHA Кодда [60], яка безпосередньо базується на реляційному обчисленні. ALPHA дозволяє користувачеві, застосовуючи такі поняття, як змінюваність і квантори, формувати непроцедурні запити. Згодом були запропоновані інші мови, що, як і ALPHA, базувалися на реляційному обчисленні. До них належать QUEL [106], створена в рамках науково-дослідницького проєкту Ingres в Каліфорнійському університеті в Берклі, COLARD (Calculus Oriented Language for Rational Data) [107], RIL [108].

Графічні мови. В мовах, котрі відносяться до графічних, формулювання запитів відбувається не з використанням традиційного лінійного синтаксису, а із заповненням осередків (ячеєк) у бланках таблиць.Мова CUPID (Casual User Pictorial Interface Design – робота з графічним інтерфейсом непрофесійного користувача) [109 - 112] надає користувачеві графічну мову запитів. CUPID має високорівневу меню – образну підмову, яка є зовнішнім інтерфейсом до системи INGRES.

Ідея мови QBE (Query by Example – запит за зразком) [113 - 117], розроблена Моше М. Злуфом (Moshe M/ Zloof), полягає в наступному. Користувачеві надаються чисті бланки таблиць бази даних. Формулювання запиту – це заповнення бланків однією правильною відповіддю, а задача системи – на підставі цього прикладу – вивести всі можливі правильні рядки таблиць. Попри очевидну простоту, було доведено [113], що QBE є реляційно повною мовою. Різновиди цієї мови були реалізовані в СУБД PARADOX, DBASE IV, ACCESS. Остання входить до складу Microsoft Office. М.М.Злуф розробив також мову OBE (Office-by-Example – офіс за зразком) [118], яка стала розширенням QBE для офісних пропозицій.

Моше М. Злуф

Мови, орієнтовані на відображення. !973 року колеги Кодда з лабораторії IBM у Сан Хосе Раймонд Бойс, Дональд Чемберлін і Вільям Кінг (William F. King) розробили мову SQUARE (Specifying Queries As Relational Expressions - специфікація запитів у вигляді реляційних виразів) [119, 120].

Використання SQUARE – подібної мови для опису численних уявлень (поглядів), а також керування цілісністю даних та їх автоматизацією описане в статті [121]. На відміну від реляційного обчислення, SQUARE не використовує кванторів та зв′язаних змінних і тому не потребує відповідної математичної підготовки. В мові запити висловлюються у вигляді природніх примітивних операцій, якими люди користуються під час пошуку інформації в таблицях. Більшість семантичних простих запитів відображаються в мові просто й лаконічно. Разом з тим SQUARE є реляційно повною мовою [120].

У 1974 році Бойс і Чемберлін представили мову SEQUEL (Structured English QUEry Language – структурована англійська мова запитів) [122], яка стала удосконаленим варіантом мови SQUARE. Змішаний синтаксис мови було названо

блочно – структурованим синтаксисом ключових слів англійської мови. SQUARE і SEQUEL були декларативними мовами. Тобто, в них формулюється «що» треба знайти, а не «як» це зробити, характерне для процедурних мов. 1975 року було реалізовано експерементальний варіант SEQUEL на базі розробленого інтерпретатора [123]. Завдання інтерпретатора – мінімізувати виконання операцій доступу до даних під час виконання запитів за рахунок звуження простору пошуку. Задля цього були досліджені спеціальні оптимізуючи алгоритми. Сам інтерпретатор SEQUEL базується на XRM (Extended n-ary Relational Memory [124, 125]) – системі, яка надає ефективний асоціативний доступ до бінарних відношень. Нарешті 1976 року було представлено мову SEQUEL2 [128], в якій уже були включені всі основні засоби для оперування базами даних: визначення, маніпулювання і керування.

Оригінальний підхід був запропонований у мові APPLE (Access Path Producing Language – мова, яка породжує шлях доступу) [129], одним з авторів якого був Карл Роберт Карлсон (Carl Robert Carlson). Мова передбачає використання в запиті тільки імена атрибутів відношень бази даних. Задача системи – на основі структури бази даних визначити численні відношення, необхідні для виконання запиту, і визначити шлях доступу до них.

Карл Роберт Карлсон

Експерементальні дослідження і розробки. Вже на початку 70-х років було реалізовано низку ранніх реляційних систем - MacAIMS (1970 р.), IS/1 (1972 р.) и PRTV, RENDEZVOUS (1974 р.) тощо.

IS/1i PRTV. IS/1 була першою в світі експериментальною реляційною системою баз даних з обмеженими можливостями, реалізованою в науковому центрі IBM у Пітерлі, Великобританія в 1970 – 1972 роках [130]. Із урахуванням результатів, отриманих під час реалізації IS/1, була розроблена СУБД PRTV (Peterlee Relational Test Vehicle) [131], що дозволяла оперувати великими обсягами даних, мала свою власну мову записів ISBL рівня реляційної алгебри і була призначена для одного користувача.

System/R і DB2. 1974 року в дослідницькій лабораторії в Сан Хосе компанії IBM був ініційований проєкт System/R зі створення експериментальної СУБД. Його задачею було продемонструвати можливість створення високопродуктивних промислових реляційних СУБД. За основу було взято мову SEQUEL, яка в процесі обробки була перейменована на SQL, виходячи з юридичних міркувань. До 1975 року було реалізовано повнофункціональну версію System/R для багатьох користувачів [132]. Зрештою, протягом 1978-79 років System/R витримала всебічну практичну апробацію [133, 134], результати якої продемонстрували, що реляційні СУБД здатні забезпечити високу продуктивність. 1979 року проект System/R було завершено. Пізніше коротка історія експериментальних досліджень проекту System/R була викладена Чемберліном і його колегами в статті [135]. Враховуючи отриманий досвід, компанія IBM 1980 року почала, а 1982 року випустила промислову реляційну СУБД під назвою SQL/DS, яка згодом була перейменована на DB2 і підтримується на сьогодні на різних платформах і в різних конфігураціях. Вона стала стратегічним програмним продуктом компанії IBM.

Oracle. 1974 року троє молодих програмістів із американської електронної компанії Ampex Corporation Ларрі Еллісон (Larry Ellison), Боб Майнер (Bob Miner) та Ед Оутс (Ed Oats), окрилені ідеями Кодда, заснували компанію Software Development Laboratories (SDL) для створення реляційного СУБД і взя-

лися за розробку і маркетинг програми. 1979 року компанія дістала нову назву – Relational Software Inc. Того ж року компанія випустила Oracle, першу комерційну реляційну СУБД, де використовувалася мова SQL. Програма дуже швидко набула популярності. 1982 року компанія знову була перейменована на Oracle Systems Corporation. Відтоді Oracle є найбільшим постачальником реляційних СУБД на базі SQL.

Ingress. 1973 року двоє вчених дослідницької лабораторії Каліфорнійського університету в Берклі Майкл Стоунбрейкер (Michael Ralph Stonebraker) і Юджин Вонг (Eugene Wong), зацікавившись дослідженнями Кодда і результатами своїх колег із IBM зі створення System R, вирішили почати власний проект зі створення реляційної СУБД.

Майкл Стоунбрейкер

Юджин Вонг

Розроблювану експериментальну СУБД було названо INGRES (Interactive Graphics and REtrieval System).

За наступні два роки були проведені експериментальні дослідження і розробки. Було прийнято проєктні рішення [136, 137], розроблені структури зберігання і методи доступу [138]. Також було розроблено оптимізаційний алгоритм виконання операцій поєднання відношень, який отримав назву алгоритм Вонга – Юсефі (Wong – Youssefi algorithm) [139], досліджено механізм надання альтернативних поглядів через підстановку в запити користувачів їхніх визначень поглядів [140]. Авторизація і контроль цілісності забезпечувався використанням додаткових предикативів до запиту користувача [141]. Реалізовано механізм безпечного одночасного оновлення бази даних [142], а також система захисту [143]. До 1976 року була реалізована експериментальна версія Ingress [144]Яка підтримувала мову QUEL. 1980 року частина співробітників цієї лабораторії заснували фірму Relational Technology, яка 1981 року випустила промислову СУБД INGRESS. 1986 року INGRESS було переведено на SQL. Кілька ключових ідей з INGRESS досі широко використовуються в реляційних системах. Наприклад, в Non Stop SQL, Sybase і Microsoft SQL Server.

Postgres. Після створення Relational Technology Стоунбрейкер разом із Лоуренсом А. Роу (Lowrence A. Rowe) почали досліджувати можливості усунення обмежень реляційної моделі.

Лоуренс А. Роу

Новий проект дістав назву Postgres (POST inGRES). Були розроблені концептуальні проєктні рішення [145], запропонована об'єктивно – реляційна модель зі складними типами даних [146], розроблені структура збереження даних [147] і система правил [148] (тригерів), яка дозволяє визначати додаткові дії, ініційовані під час виконання операцій вставки, оновлення або видалення в таблицях бази даних.

Попервах мовою запитів Postgres була PostQUEL. Мова була розроблена 1985 року в Каліфорнійському університеті в Берклі під керівництвом М. Стоунбрейкера. PostQUEL базувалася на мові запитів QUEL. 1987 року була реалізована перша версія СУБД Postgres, яка протягом наступних кількох років удосконалювалась [149]. Postgres почала широко використовуватись в економіці, промисловості, медицині, фінансовій справі, астрономії і в багатьох інших галузях. А також використовувалася в навчальному процесі. 1991 року було додано інтерпретатор мови SQL, а 1996 – програмний продукт було перейменовано на PostgreSQL. 2014 року Майкл Стоунбрейкер став лауреатом премії Тьюринга за фундаментальний внесок у концепції і методи, що лежать в основі сучасних систем баз даних [150].

СУБД для ПК. До 1980 року дослідження, експериментальні й промислові розробки СУБД проводилися для великих і середніх комп′ютерів. На початку 80-х з′явилися IBM PC і сумісні з ним ПК, оснащені ОС MS-DOS, що привело до появи СУБД для ПК. 1981 року компанія Ashton-Tale випустила dBase II для ПК. ЇЇ не можна було назвати справжньою СУБД, бо чимало важливих функцій не підтримувались, однак для ПК того часу це було значною подією. Dbase II здобула велику популярність. 1984 року було випущено досконалішу версію dBase III, в 1986 – її розширений варіант dBase III+, а 1998 - dBase IV. Вони стали домінуючими СУБД для IBM PC. Успіх dBase III+ обумовив появу на ринку числених аналогів, сумісних за мовою і структурою файлів бази даних. До них відносяться FoxBASE (1984), FoxPro (1990) компанії Fox Software, Clipper (1985) компанії Nantucker Corporation. Згодом вони були об′єднані популярним серед професіоналів поняттям "xBase". Тенденція створення продуктів – аналогів і велика популярність xBase активізувала діяльність зі створення стандарту. Було зроблено дві спроби стандартизації мови xBase у 1987 – 1988 і 1992 роках, але вони не мали успіху. Тож у 80-х роках домінуючу роль на ринку СУБД для IBM PC мало сімейство СУВД xBase.

1985 року компанія Ansa Software випустила СУБД Paradox. Цей високоефективний продукт для створення реляційних баз даних був примітний своєю мовою QBE (Query By Example) і мовою розробки додатків. Він був популярний у кінці 80-х – початку 90-х років і конкурував із сімейством xBase.

Оптимізація. Реляційні системи базуються на високорівневому непроцедурному інтерфейсі, їхні мови записів декларативні. Через це в таких системах принципово важливим є питання оптимізації виконання запитів. У 70 – 80 роки минулого століття були проведені численні дослідження і опублікована величезна кількість статей із цього питання. Ми не зупиняємося в цій статті на даній проблемі і посилаємо читача до змістовного огляду С.Д.Кузнєцова [151].

Стандартизація. У травні 1979 року була створена робоча група із реляційних баз даних (RTG) ANSI/X3/SPARC DBS-SG під керівництвом Майкла Броуді (Michael L.Brodie) для проведення досліджень з обґрунтування можливості створення стандарту по реляційних базах даних. 1981 року ця група склала звіт [152], де підтверджувалась така необхідність. Для наступного сприяння в роботі зі створення такого стандарту було розроблено «Каталог функцій реляційних концепцій, мов і систем», який мав би допомогти виявити і встановити ті аспекти, як самої реляційної моделі, так і реляційних баз даних, які можуть розглядатися кандидатами для стандартизації. До початку 80-х років у зв'язку із широким розповсюдженням реляційних СУБД виникла необхідність аналізу можливості стандартизації мови для управління

реляційними базами даних і розробки такого стандарту, якщо це буде визнано доцільним.

Майкл Броуді

У зв'язку з цим 1982 року Американський національний інститут стандартів (American National Standards Institute – ANSI) створив комітет X3H2, перед яким було поставлене це завдання. Впродовж 11 років комітет очолював Дональд Р. Дойч (Donald R.Deutsch). Комітет мав розглянути різні реляційні мови, описані і реалізовані на той час. Однак, враховуючи широку розповсюдженість SQL у промислових СУБД і той факт, що тоді він фактично вже став стандартом, комітет зупинив свій вибір саме на цій мові. Узявши за основу її діалект, реалізований в СУБД DB2, комітет прагнув його узагальнити, враховуючи реалізовані в інших реляційних СУБД можливості. Після чотирьох років роботи, 1986 року запропонований комітетом варіант SQL, було офіційно затверджено як стандарт ANSI, а 1987 року він був узятий за стандарт Міжнародної організації стандартів (International Standards Organization – ISO). Згодом стандарт ANSI/ISO взяв уряд США за федеральний стандарт в галузі обробки інформації (Federal Information Processing Standard – FIPS). 1989 року стандарт був дещо змінений і дістав назву SQL – 89 (або SQL1).

Відтоді SQL було визнано єдиною мовою зовнішніх інтерфейсів реляційних баз даних. ANSI/ISO постійно працює над її удосконаленням і випуском нових версій. За 35 років було випущено 10 версій SQL (1986, 1989, 1999, 2003, 2006, 2008, 2011, 2016, 2019).

Дональд Р. Дойч

На завершення цього розділу зазначимо, що до початку 80-х років на ринку з'явилися дві промислові реляційні СУБД: Oracle і DB2. Згодом з'явилися Postgress, Informix тощо. Почалася ера реляційних СУБД, які й по сьогодні є найбільш популярними на ринку баз даних.

Управління транзакціями.

Важливою функцією використання БД є управління транзакціями. Транзакція (Transaction) – це логічна одиниця роботи, що являє собою групу послідовних операцій над даними БД, яка може бути або використана повністю й успішно, дотримуючись цілісності даних і незалежно від інших паралельно працюючих транзакцій, або не використана зовсім. Тоді вона не матиме ніякого ефекту. Поняття транзакції вперше було введене й обговорене Джимом Греєм (Jim Grey) і його колегами в працях [153, 155, 158].

Правила ACID. Одним із найпоширеніших наборів вимог до транзакцій і транзакційних систем є набір ACID (Atomicity, Consistency, Isolation, Durability).

- *Atomicity – атомарність.* Транзакція або виконується повністю, або не виконується взагалі. З точки зору зовніш-

нього сприйняття, вона не має ніяких проміжних станів.

- *Consistency – узгодженість.* Транзакція зберігає обмеження цілісності бази даних. По завершенні роботи транзакція залишає базу даних у цілісному стані.
- *Isolation – ізольованість.* Транзакція працює так, ніби не було ніяких одночасно працюючих транзакцій.
- *Durability –довготривалість.* По закінченні роботи всі зроблені транзакцією зміни зберігаються в базі даних на довготривалій основі.

Вимоги ACID були сформульовані переважно на початку 80-х років Джимом Греєм [153]. Водночас існують спеціальні системи з послабленими транзакцій ними властивостями [154].

Ізольованість транзакції – ситуація, в якій транзакція захищена (за ізольована) від дії інших одночасно з нею виконуваних транзакцій. Іншими словами, ізоляція гарантує, що проміжні стани транзакції є невидимими іншим одночасно працюючим транзакціям. Ступінь ізоляції транзакції визначається рівнями ізоляції. Аби досягти ізольованості транзакції, слід використовувати методи управління спільним виконанням транзакцій. План виконання набору транзакцій називається серіальним у разі, якщо результат спільного виконання транзакцій еквівалентний результату певного послідовного виконання цих таки транзакцій.

Серіалізація. Серіалізація транзакцій – це механізм такого спільного виконання транзакцій, коли результат еквівалентний результату певного послідовного виконання цих транзакцій. Забезпечення такого механізму є основною функцією управління транзакціями. Система, в якій підтримується серіалізація транзакцій, забезпечує реальну ізольованість користувачів.

Концепція серіалізації була сформульована і досліджена Греєм і його колегами в працях [155, 158]. Окрім цього, в праці [6] було визначено двофазний протокол блокування, а також досліджена техніка предикатного блокування. Питання межі дії блокування (гранулярності) обговорені в статтях [155, 156].

Моделі транзакцій. Варто зазначити, що всі моделі транзакцій визначалися, як правило, з урахуванням класів прикладних систем, де вони застосовуються [11]. Було запропоновано дві фундаментальні моделі транзакцій – модель сторінки і модель об′єкту. Перша з них – виконавча модель, а друга – концептуальна.

Модель сторінок (модель Read/Write). Базується на припущенні, що основні операції бази даних – це запис і читання сторінки, котрі передаються між зовнішньою і оперативною пам′яттю. Сторінкова модель транзакції бере свій початок у другій половині 70-х років зі статей Джима Грея [153, 157] і Капалі Есваран (Kapali Eswaran) [158]. Одночасно виникло споріднене поняття – атомарна дія [159, 160]. Концепція сторінкової моделі транзакції стала предметом інтенсивних теоретичних досліджень 80-х років [161 - 164] і дієва дотепер, хоча й набула кількох розширень і варіацій [165]. Із оглядом досліджень і розробок цієї моделі можна ознайомитися в працях [11, 166].

Усі наведені далі моделі належать до класу так званих моделей об′єктів.

Плоскі транзакції (Flat Transactions) мають єдиний рівень управляння для довільної кількості елементарних дій. Вони не мають внутрішньої структури. Плоскі транзакції – основні будівельні блоки для реалізації принципу атомарності. В плоских транзакціях атомарність і довготривалість підтримуються механізмом відновлення, який зазвичай забезпечується веденням журналів операцій оновлення, через що операції типу «відмінити», «повторити» можна виконувати за потреби. Ізольованість забезпечується механізмом управління паралелізмом (concurrency control), який реалізується з допомогою блокувань.

Огляд досліджень з управління паралелізмом наведено в праці [167]. Узгодженість забезпечується механізмом управління цілісністю. Було запропоновано два підходи до управління цілісністю в транзакціях: включення цього механізму в СУБД [168] і підтримання цілісності за рахунок зусиль розробників додатків [11].

Плоскі транзакції повністю дотримуються всіх принципів ACID і є цілком достатніми для багатьох традиційних додатків баз даних, у яких час виконання транзакції відносно нетривалий, кількість паралельних транзакцій досить невелика, і база даних не розподілена. Однак такі ACID-транзакції не взмозі підтримувати довгочасні транзакції та транзакції зі складною внутрішньою структурою і розподіленими базами даних.

Точки збереження. Це такі моменти в обчислювальному процесі, починаючи з яких можливий перезапуск обчислень у разі виникнення будь-яких проблем. Вони вперше були визначені 1976 року в SystemR [132]. У разі виникнення збою відбувається відкат до останньої збереженої точки з вивільненням усіх зроблених після цієї точки блокувань. Хоча механізм точок збереження широко використовується в плоских транзакціях, однак він набув нового звучання в розширених моделях транзакцій, які з′явилися у 80-х роках.

Усі наведені далі розширені моделі транзакцій призводять до ослаблення тих чи інших складових ACID.

Модель багатоланкових транзакцій (Chained Transactions) подібна до моделі плоскої транзакції з точки збереження, але вона не лише дає можливість позначити будь-яку точку для можливого повторного виконання, а й фіксацію тієї частини роботи, яка була виконана в момент досягнення цієї точки. Втім відкат може бути виконаний лише до останньої контрольної точки. В цьому підході було закладено ідею декомпозиції великих транзакцій на дрібніші послідовно виконувані субтранзакції, які відповідають інтервалам між точками. У разі збою поточної субтранзакції, попередня вже була зафіксована і її результати були збережені в базі даних, тому відкат проводиться до цієї точки збереження. Зазначимо, що в цій моделі атомарність і ізольованість не гарантуються для всієї транзакції. Згідно [11] ідея багатоланкової транзакції вперше реалізована в системі IMS компанії IBM.

Вкладені транзакції (Nested Transactions). Важливим кроком у розвитку базової моделі транзакції було розширення плоскої (однорівневої) моделі в багаторівневу структуру. Вкладена транзакція вперше була визначена 1981 року Моссом (Moss) [169], а потім [170]. Її концепція базувалася на понятті сфер контролю (spheres of control) [171]. Вкладена транзакція – це численні субтранзакції, що здатні містити інші субтранзакції, таким чином утворюючи транзакційне дерево. Дочірні транзакція запускається після батьківської, а батьківська транзакція завершується лише після завершення роботи всіх її дочірніх транзакцій. У разі аварійного завершення батьківської транзакції, всі її дочірні транакції також завершуються аварійно. У разі аварійного завершення дочірньої транзакції її батько може обрати альтернативний варіант (contingency subtransaction – транзакція на непередбачуваний випадок). Вкладені транзакції забезпечують повну ізоляцію на глобальному рівні. Для вкладених транзакцій послаблена здатність довготривалості ACID.

Модель вкладених транзакцій повністю відповідає активним базам даних, оскільки ієрархічна структура моделі дозволяє легко погоджувати зв′язок між основною трансакцією і запущеною з допомогою тригера. В статті [172] запропоновані наступні варіанти синхронізації запуску субтранзакції тригеру і щодо основної транзакції:

- негайний (immediate) - транзакція запускається одразу після настання події тригера;
- відкладається (deffered) - запуск субтранзакції відкладається до завершення основної транзакції;
- причинно-незалежна (causally independent) – субтранзакція тригера запускається як цілком самостійна транзакція;
- причинно-залежна (causally dependent) – субтранзакція тригера запускається як самостійна транзакція, однак її успішне завершення залежить від успішного завершення основної транзакції.

Відкриті вкладені транзакції (Open Nested Transactions) [173] послаблюють вимогу ізольованості через те, що результати зафіксованих субтранзакцій стають видимими іншим одночасно працю-

ючим вкладеним транзакціям. При цьому досягається високий рівень паралельності.

Багаторівневі транзакції є найзагальнішим варіантом вкладених транзакцій [173, 174]. Субтранзакції багаторівневої транзакції здатні провести фіксацію і звільнення своїх ресурсів до завершення роботи глобальної транзакції. Якщо глобальна транзакція завершується аварійно, то для підтримки атомарності слід зробити відкат субтранзакції запуском їх компенсуючи субтранзакцій. Однак все ж можливе порушення цілісності в зв′язку з тим, що певна інша транзакція мала доступ до результатів завершених субтранзакцій, які потім були відкатані компенсуючими субтранзакціями. Було запропоновано вирішення цієї ситуації. Зокрема, введенням горизонтальних компенсаторів [175]. У монографії [176] наводяться розпізнавальні ознаки багаторівневих і вкладених транзакцій.

Розподільні транзакції (Distributed Transactions). Це сукупність субтранзакцій, прив′язаних до локальних баз даних і загальної глобальної транзакції. У статті [177] подається огляд розподілених транзакцій, а також розглянута «модель базисної транзакції» (Base Transaction Model) і її розширення. 1996 року було запропоновано модель x/Open Distributed Transaction Processing (x/Open DTP) [178]. Ця модель є стандартом для протоколу двофазної фіксації (2PC – Two Phase Commit).

Гнучкі транзакції (Flexible Transactions) [179, 180] – були запропоновані для розподілених баз даних. У цьому випадку глобальна транзакція є набором субтранзакцій, кожна з яких здійснює доступ до даних на одному локальному вузлі. Модель гнучкої транзакції підтримує гнучке управління обчисленням шляхом специфікації залежностей двох типів між субтранзакціями: 1) залежності порядку обчислень між двома субтранзакціями; 2) залежності альтернатив між двома піднаборами субтранзакцій. Було розроблено кілька конкретних моделей гнучких транзакцій: Con Tracts, Flex Transaction, Split Transaction, S-transaction та інші [179 - 183]. Була також запропонована мова IPL [184] для специфікації гнучких транзакцій із атомарністю й ізольованістю, які визначаються користувачем.

Тривалі транзакції, компенсатори і модель Saga. Ідея компенсуючих транзакцій (Compensation Transactions) вперше була висловлена Греєм у праці [157], згодом вона була формалізована в [185, 186] і, зрештою, використана в моделі Saga [187]. Ця модель належить до тривалих транзакцій (Long – Running Transactions) [188]. Моделі розподільних транзакцій вдало справляються з короткочасними транзакціями, однак, водночас є неприйнятними для тривалих транзакцій. В моделі Saga пропонується розподіляти тривалі транзакції на коротші. Saga складається із сукупності впорядкованих ACID субтранзакцій і сукупності компенсуючих субтранзакцій, по одній на кожну з основних субтранзакцій. Координація всього процесу здійснюється за допомогою повідомлень і тимчасових поміток. Saga завершується успішно за умови успішної фіксації всіх субтранзакцій. Якщо ж котрась із субтранзакцій завершується аварійно, то всі попередньо завершені субтранзакції відкочуються виконанням так званих компенсуючи субтранзакцій. Saga послаблює вимоги до ізольованості і збільшує міжтранзакційний паралелізм. У праці [189] запропонована вдосконалена модель – вкладена Saga, яка дозволяє представляти лінійну структуру тривалих транзакцій у вигляді ієрархічної транзакційної структури.

Транзакції Split/Joint (розділити/об′єднати). Концепція роз′єднання/об′єднання транзакцій вперше була описана в [190], а відтак ретельно пропрацьовано в [191] для таких тривалих видів діяльності, як автоматизоване проєктування, інженерне проєктування, проєктування і розробка програмного забезпечення тощо. Операція Split розділяє транзакцію на дві серіалізовані транзакції, які фіксуються або завершуються аварійно незалежно одна від одної. Операція Joint об′єднує дві транзакції в одну. Split використовується, наприклад, щоб якнайшвидше зафіксувати результати роботи частини транзакцій, або щоб розподілити роботу між кількома виконавцями. Зі свого боку Joint рівно-

сильний передачі всієї роботи одному виконавцеві [191]. Згодом операції Split та Joint було включено до вкладених транзакцій для створення комбінованих моделей транзакцій [192].

Кооперативні транзакції були запропоновані в [193] для використання в системах, де яскраво виражена потреба у взаємодії між транзакціями, в кооперативному інтерактивному робочому середовищі. Фундаментальна проблема, пов′язана з кооперативними транзакціями – це відсутність для них чітких критеріїв узгоджуваності. Для кооперативних транзакцій було запропоновано їхню структуризацію у вигляді дерева, названого ієрархією кооперативних транзакцій (Cooperative Transaction Hierarchy). В окремому випадку ієрархія обмежена трьома рівнями: корінь, одна або більше транзакційних груп і кілька кооперативних транзакцій. Кооперативні транзакції утворюють листя ієрархії, яке об′єднується в групи. Члени групи транзакцій працюють разом, виконуючи певну логічну одиницю роботи, названу задачею, котра може бути розбита на підзадачі. Кожна кооперативна транзакція відповідальна за конкретну підзадачу. Оскільки вимога атомарності послаблена, кооперативна транзакція не зобов′язана зберігати глобальне непротиріччя бази даних, тобто зміни, зроблені кооперативною транзакцією, одразу стають видимими іншими кооперативними транзакціями цієї групи. Щоб результати були видимі поза групою, використовуються точки збереження. В ієрархії може існувати більше трьох рівнів, тобто допускається кілька рівнів вкладення груп. Кооперативні транзакції не обов′язково мають бути серіалізованими. Через свою інтерактивну природу, кооперативні транзакції тривають значно довше звичайних. Згодом, на відміну від традиційних транзакцій, кооперативні не обов′язково мають бути повністю ізольованими.

ACTA і її похідні. ACTA [192, 194, 195] – це мета модель, яка полегшує специфікацію, аналіз і синтез розширених моделей транзакції. Формалізм ACTA має за основу логіку першого порядку з відношенням перебування, яке дозволяє розробнику транзакцій специфікувати як високорівневі властивості (вимоги) моделі, так і низькорівневі аспекти поведінки в термінах аксіом. Окрім підтримки специфікації і аналізу існуючих моделей транзакцій, ACTA дає можливість специфікувати вимоги нових транзакцій них додатків та синтезувати моделі, що задовольняють ці вимоги. В роботі [42] автори запропонували спрощений спосіб розробки розширених транзакцій, названий ASSET. Він базується на транзактивних примітивах, запозичених у ASTA і може використовуватися на рівні програмування для специфікації спеціалізованих для конкретних додатків моделей транзакцій, які дозволяють підтримувати кооперацію і взаємодію. У 90-і роки велика увага приділялася транзакціям в системах реального часу [197, 198, 199] і в мобільних системах баз даних [200, 201].

Транзакції веб – сервісів. Із початком 2000-их років усе більша увага приділяється використанню транзакцій для слабопов′язаних веб-сервісів із метою забезпечення погоджуваності й надійності веб-сервісних додатків. Наразі розроблено три стандарти, що мають відношення до транзакцій веб-сервісів.

Business Transaction Protocol (BTP) [202, 203]. Перша версія розроблена 2004 року в OASIS. Вона стосується як веб-сервісів, так і довільних бізнес процесів. Це протокол, що базується на мові XML, для опису і управління складними багатокроковими B2B-транзакціями (business-to-business) в Інтернеті. Він дає можливість координувати транзакції між багатьма автономними сервісами, а використання XML робить його придатним для веб-сервісних архітектур [204].

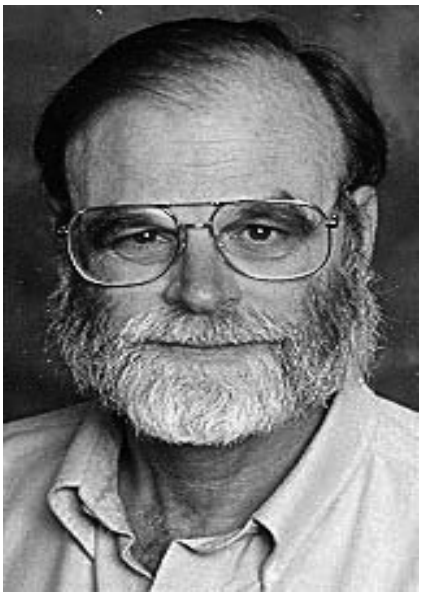
Web Services Transactions (WS – Tx) [205, 206]. Специфікація, схвалена 2007 року, складається з WS – Coordination (WS-C), WS-Atomic Transaction (WS – AT), WS-Business Activity (WS – BA) та розроблена в Microsoft, IBM і BEA. WS – Tx визначає механізми транзакційної інтероперабельності між веб-сервісами та забезпечує впровадження якісних транзакцій них сервісів до веб=сервісних додатків. WS – Tx визначає послідовність

повідомлень, що передаються сторонами – учасниками в короткострокових атомарних транзакціях (WS – AT) і (тривалих) бізнес – транзакціях (WS – BA). WS – C визначає координаційні протоколи повідомлень, якими обмінюються сторони – учасники транзакцій. WS – C підтримує різні координаційні моделі [2007].

WS Composite Application Framework (WS – CAF) [2008]. Стандарт розроблено в OASIS за участі компаній SUN, Oracle, Arijuna та інших. Мета стандарту – розробка інтероперабельних і простих у використанні складових вебсервісних додатків.

Було здійснено порівняльні дослідження наведених вище трьох стандартів, з результатами яких можна ознайомитися в статтях [209, 211].

1998 року Джеймс Ніколас Грей (James Nicholas Gray) був нагороджений премією Тьюрінга за основоположні ідеї в галузі баз даних, за дослідження з обробки транзакцій і технічне лідерство в реалізації систем.



Джеймс Ніколас Грей

(Далі буде)

References

1. Olle T. William. The CODASYL Approach to Data Base Management. Chichester, England: Wiley-Interscience; 1978: 287p.
2. Tsichritzis D.C., Lochovsky F.H., Data models, Prentice-Hall, Englewood Cliffs, N.J., 1982, 381 p.
3. Embley D., Thalheim B., editors. Handbook of conceptual modelling: its usage and its challenges. Springer; Berlin 2011
4. Date C.J. An Introduction to Database Systems, 8th Edition. Addison-Wesley Longman Publishing Co., Inc.75 Arlington Street, Suite 300 Boston, MA United States
5. Gallaire H., Minker J., eds. Logic and Databases. New York: Plenum. 1978.
6. Ullman J.D. Principles of Database and Knowledge-Based Systems.Maryland: Computer Sciences Press Inc., 1989
7. Maier D., Warren D.S. 1988. Computing with Logic: Logic Programming with Prolog. Benjamin-Cummings Publishing Co., Inc.Subs. of Addison-Wesley Longman Publ. Co390 Bridge Pkwy. Redwood City, CA United States. 535 p.
8. Ozkarahan E.A. Database Machines And Database Management. Prentice Hall, 1986, 636 p.
9. Kalinichenko L.A., Ryvkin V.M. Database and Knowledge base Machines (Rus). Moscow, Nauka, 1990, 296 p.
10. Özsu M.T., Valduriez P. Principles of Distributed Database Systems, Fourth Edition, Springer, 2020
11. Gray J., Reuter A. Transaction Processing: Concepts and Techniques. Morgan Kaufmann, San Francisco. 1993
12. Harrison G. Next Generation Databases: NoSQL, NewSQL and Big Data, Apress, 2015, 235 p.
13. Kogalovsky M. R. Encyclopedia of databases technologies (Rus). Moscow, Finance and Statistics,.2005. — 800 c.
14. Bachman Charles W. Integrated Data Store - The Information Processing Machine That We Need! General Electric Computer Users Symposium. Kiamesha Lake. New York May 17-18, 1962
15. IDS Reference Manual GE 625/635, GE Inform. Sys. Div., Pheonix, Ariz., CPB 1093B, Feb. 1968.
16. Bachman Charles W. “The Origin of the Integrated Data Store (IDS): The First Direct-Access DBMS,” IEEE Annals of the History of Computing, Vol. 31, Num. 4, Oct-Dec 2009, pp. 42-54.

17. History of IMS: Beginnings at NASA. - https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.imsintro.doc.intro/ip0ind0011003710.htm#ip0ind0011003710
18. Long R., Harrington M., Hain R., Nicholls G. IMS Primer. - http://www.redbooks.ibm.com/redbooks/pdfs/sg245352.pdf
19. Information Management System/360, Application Description Manual H20-0524-1. IBM Corp., White plains, N.Y., July 1968.
20. Nies T. Cincom Systems' Total. Annals of the History of Computing, IEEE. 2009, vol. 31, No 4, pp. 55-61.
21. Bachman Charles W. "The programmer as navigator". Communications of the ACM, November 1973, Vol. 16 No. 11, Pages 653-658 https://dl.acm.org/doi/10.1145/355611.362534
22. Haigh T. How Data Got its Base: Information Storage Software in the 1950s and 1960s // IEEE Annals of the History of Computing (Volume: 31, Issue: 4, Oct.-Dec. 2009) pp. 6-25
23. CODASYL: "Data Base Task Group Report", ACM (New York 1971).
24. GUIDE-SHARE: "Data Base Management System Requirements", SHARE Inc. (New York 1970)
25. CMSAG Joint Utilities Project: "Data Management System Requirements", CMSAG (Orlando, FL 1971)
26. Langefors B. Theoretical Analysis of Information Systems. 402 S. m. Fig. Lund/Kopenhagen/Oslo 1966. Akademisk Forlag/Universitetsforlaget
27. Langefors B. Information systems theory. Inf. Syst. 2(4): 207-219 (1977)
28. Langefors B. Infological models and information user views. Information Systems Volume 5, Issue 1, 1980, Pages 17-32
29. Sungren Bo. An Infological Approach to Data Bases. National Central Bureau of Statistics, Sweden, Stokholm, 1973. 294 p.
30. SPARC: "Outline for Preparation of Proposals for Standardization", Document SPARC/90, CBEMA (Washington, DC 1974).
31. ANSI/X3/SPARC, 'Study Group on Data Base Management Systems: Interim Report 75-02-08' // Newsletter ACM SIGMOD Record, FDT, Vol 7, No. 2, 1975. – P. 1-140
32. Tsichritzis D.C., Klug A. "The ANSI/X3/SPARC DBMS Framework". Report of the Study Group on a Database Management System". Information Systems, Vol. 3, No. 4, 1978.
33. CODASYL/Data Description Language Committee (DDLC), "June 73 Report". CODASYL Data Description Language Committee Journal of Development, June 1973
34. "CODASYL Data Description Language Committee Journal of Development", 1978.
35. Concepts and Terminology for the Conceptual Schema and the Information Base, van Griethauzen, J.J., Ed., ISO TC97/SC5/WG3, 1982, Publ. 695.
36) Falkenberg E,D. Structuring and Representation of Information at the Interface Between Data Base User and Data Base Management System. Diss. Univ. Stuttgart (1975).
37. Falkenberg E., Concepts of Modelling Information, Proc. of the IFIP Working Conf. on Modelling in Data Base Management Systems, Nijssen, G.M., Ed., North-Holland, 1976, p. 95-109.
38. Abrial Jean-Raymond, Data Semantics, In: J. W. Klimbie, K. L. Koffeman (eds.), Database Management, Proceedings IFIP TC2 Conference. Grgese, 1974., North-Holland Publishing Company, pp.1-60.
39. Bracchi G., Paolini P., Pelagatti G. "Binary Logical Associations in Data Modelling," in J. M. Nijssen (ed.), Modelling in Database Management Systems (Proc. IFIP TC2 Conference, Freudenstadt), North-Holland Publishing Company, Amsterdam, The Netherlands, 1976.
40. Durchholz R. and Richter G., "Concepts for data base management systems". In: Data Base Management, J. W. Klimbie and K. L. Koffeman, (eds.),
41. Senko, M.E., Conceptual Schemas, Abstract Data Structures, Enterprise Descriptions, In: International Computing Symposium, Liege, Belgium, 1977, North-Holland Publishing Company.
42. Senko M.E., Altman E.B., Astrahan MM., Fehder P.L. Data Structures and Accessing in Data-Base Systems. IBM System J., v. 12, no. 1 (1973).
43. Senko M.E., "The DDL in the Context of Multilevel Structured Description: DIAM II with FORAL". Proc. of the IFIP TC-2 Spe-

cial Working Conference on Data Base Description, pp.239-257, Jan. 1975
44. Smith J.M. and Smith D.C.P. Databas e Abstractions : Aggregation and Generalization. ACM Trans, on Database Syst, v. 2, no. 2, 1977, pp. 105133
45. Smith J.M. and Smith D.C.P. Databas e Abstractions: Aggregation. Comm. of the ACM, v. 20, no. 6, 1977, pp. 405-413
46. Hammer, M. and McLeod, D., Database Description with SDM: A Semantic Database Model, ACM Transactions on Database Systems,1981, Vol. 6, No. 3, pp. 351-386.
47. Abiteboul, S., Hull, R., IFO: A Formal Semantic Database Model, ACM Trans. Database Syst. 12, 4 (1987), 525-565.
48. Bachman, C. W., "Data Structure Diagrams", Data Base, 1969, No 1, 2, pp. 4-10.
49. Engles R.W. A Tutorial on Data-base Organization, Annual review in automatic programming, Vol 7. Part I, Pergamon Press, 1972, 93 p.
50. Chen P.P. The Entity-Relationship Model — Toward a Unified View of Data // ACM Transactions on Database Systems (TODS), 1976. — Vol. 1, No. 1. — P. 9–36.
51. Barker R. Case*Method: Entity Relationship Modelling Publisher: Addison-Wesley, 1990, 240 p.
52. Gogolla M. An extended entity-relationship model – fundamentals and pragmatics. LNCS, vol. 767. Berlin: Springer; 1994.
53. Hartmann S. Reasoning about participation constraints and Chen's constraints. In: The Fourteenth Australian Database Conference, Adelaide, Australia. Conferences in Research and Practice in Information Technology; 2003. p. 105–113.
54. Hohenstein U. Formale Semantik eines erweiterten Entity-Relationship-Modells. Stuttgart: Teubner; 1993.
55. Thalheim B. Entity-relationship modeling – foundations of database technology. Berlin: Springer; 2000.
56. Teorey, T.J., Yang, D. and Fry, J.P. A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model, ACM Computer Surveys, 1986, Vol.18, No. 2. pp. 197-222
57. Vincent S. Lai, Jean Pierre Kuilboer, Jan Lucille Guynes. Temporal databases: model design and commercialization prospects. ACM SIGMIS Database: the DATABASE for Advances in Information Systems, 1994, Vol. 25, No 3, pp. 6-18
58. Gregersen H., Jense C.S. Temporal Entity-Relationship Models—a Survey. IEEE Transactions on Knowledge and Data Engineering, 1999, Vol. 11, No. 3, pp. 464 - 497
59. Codd E.F. "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, Vol. 13, No. 6 (June 1970), pp 377-397
60. Codd E.F. "A data base sublanguage founded on the relational calculus," Proc. 1971 ACM-SIGFIDET Workshop on Data Description, Access, and Control, Nov. 1971. ACM. New York, 1971, DP. 35-68
61. Codd E. F. "Relational Completeness of Data Base Sublanguages" (presented at Courant Computer Science Symposia Series 6, "Data Base Systems." New York City, N.Y. May 24th-25th. 1971), IBM Research Report RJ987
62. Palermo F.P. "A data base search problem", Proceedings 4th Computer and Information Science Symposium (COINS IV), Miami Beach, Dec. 1972, Plenum Press, New York, 1972. pp. 67–101
63. Codd E.F. "Interactive Support for Nonprogrammers: The Relational and Network Approaches,"Proceedings of the ACM SIGMOD Workshop on Data Description, Access, and Control, Vol. II, Ann Arbor, Michigan, May 1974.
64. Codd E.F. Extending the database relational model to capture more meaning. ACM Trans. on Database Syst., vol. 4, No. 4, 1979, pp. 397-434
65. Codd E. F. "The Second and Third Normal Forms for the Relational Model", IBM technical memo (October 6th. 1970).
66. Codd E.F. "Further Normalization of the Database Relational Model", in Data Base Systems, Courant Inst. Comput.Sci. Symp. Series 6 (New York, 1971), Englewood Cliffs, N.J.: Prentice Hall, 1972, pp. 33-64.
67. Codd E.F. "Normalized Data Base Structure: A Brief Tutorial", Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access. and Control. San Diego. Calif. 1971, p. 1-17
68. Date C. J. The database relational model : a retrospective review and analysisю - Ad-

dison-Wesley Educational Publishers Inc., 2000, 152 p.
69. Codd E.F. "Recent Investigations in Relational Database Systems," Information Processing 74, pp.1017–1021.
70. Heath I.J. Unacceptable File Operations in a Relational Data Base. Conference: Proceedings of 1971 ACM-SIGFIDET Workshop on Data Description, Access and Control, San Diego, California, November 11-12, 1971, pp. 19–33
71. Armstrong William Ward. "Dependency structures of data base relationships". In Jack L. Rosenfeld and Herbert Freeman, editors, Proceedings of IFIP Congress 74, pp.580-583, North Holland, 1974
72. Fagin R. Multivalued Dependencies and a New Normal Form for Relational Databases / R. Fagin // ACM Transactions on Database Systems. – 1977. – Vol. 2, № 1. – P. 262-278.
73. Beeri C., Fagin R., Howard J.H. A complete axiomatization for functional and multivalued dependencies in database relations. Proc. ACM SIGMOD Conf., D.C.P. Smith, Ed., Toronto, Canada, August 1977, pp. 47-61.
74. Zaniolo C. Analysis and design of relational schemata for database systems. Ph.D. Diss., Tech. Rep. UCLA-ENG-7669, U. of California, Los Angeles, Calif., July 1976.
75. Delobel C., Leonard M. The decomposition process in a relational model. Proc. Int. Workshop on Data Structure Models for Information Systems, Presses U. de Namur, Namur, Belgium, May 1974, pp. 57-80.
76. Rissanen J. Theory of relations for databases--A tutorial survey, in "Proc. 7th Sympos. on Math. Found. of Computer Science," 1978, pp. 537-551, Lecture Notes in Computer Science No. 64, Springer-Verlag, Berlin
77. Fagin R. Normal Forms and Relational Database Operators / R. Fagin // Proceedings of the ACM SIGMOD International Conference on Management of Data (Boston, Mass., May 30-June 1), ACM, New York, 1979, p. 153-160
78. Date Chris J. "On DK/NF normal form". - https://web.archive.org/web/20120406123712/http://www.dbdebunk.com/page/page/621935.htm
79. Buy B., Puzikova A. V. Some nonclassical normal forms in relational databases (Rus) // Bulletin of Taras Shevchenko National University of Kyiv. Series Physics & Mathematics, 2015, No 1, pp. 65-74
80. Ling T. W. An Improved Third Normal Form for Relational Databases / T. W. Ling, F. W. Tompa, T. Kameda // ACM Transactions on Database Systems. – 1981. – Vol. 6, № 2. – P. 329-346.
81. Zaniolo C. A New Normal Form for the Design of Relational Database Schemata / C. Zaniolo // ACM Transactions on Database Systems. – 1982. – Vol. 7, № 3. – P. 489-499
82. Normann R. Minimal lossless decompositions and some normal forms between 4NF and PJ/NF / R. Normann // Information Systems. – 1998. – Vol. 23, № 7. – P. 509-516.
83. Vincent M. W. A corrected 5NF definition for relational database design / M. W. Vincent // Theoretical Computer Science (TCS). – 1997. – Vol. 185, № 2. – P. 379-391.
84. Vincent M.W. Redundancy Elimination and a New Normal Form for Relational Database Design / M. W. Vincent // In Semantics in Databases (Libkin, L., Thalheim, B., eds.), vol. 1358 of LNCS. – 1998. – P. 247-264.
85. Darwen H. A Normal Form for Preventing Redundant Tuples in Relational Databases / H. Darwen, C. Date, R. Fagin // Proceedings of the 15th International Conference on Database Theory – ICDT'2012, March 26– 30, 2012, Berlin, Germany. – P. 114-126.
86. Fagin R. A Normal Form for Relational Databases That Is Based on Domains and Keys / R. Fagin // Communications of the ACM. – 1981. – Vol. 6. – P. 387-415.
87. Delobel C. Normalization and hierarchical dependencies in the relational data model. ACM TODS 1978, 3, 3, 201-222.
88. Pasichnik V.V., Stogniy A. A. Relational models of data bases (Rus). - M.: CNIIATOMINFORM, 1983, 268 p.
89. Casanova M.A. Inclusion dependencies and their interaction with functional dependencies / M. A. Casanova, R. Fagin, C. H. Papadimitriou // Journal of Computer and System Sciences. – 1984. – № 28. – P. 29-59.
90. Nicolas J.M. Mutual dependencies and same results on indecomposable relations / J. M. Nicolas // Proceedings of the fourth interna-

tional conference on Very Large Data Bases, 1978. – Vol. 4. – P. 360-367.
91. Ling T.W. Logical Database Design with Inclusion Dependencies / T. W. Ling, C. H. Goh // In Proceedings of the Eighth International Conference on Data Engineering, Tempe, Arizona, 1992. – P. 642-649.
92. Levene M. Justification for Inclusion Dependency Normal Form / M. Levene, M. W. Vincent // IEEE Transactions on Knowledge and Data Engineering, 2000. – Vol. 12, № 2. – P. 281-291.
93. Thalheim B. Bibliographie zur Theorie der Abhangigkeiten in relationalen Datenbanken, 1970-1984, TU Dresden 566/85, Dresden 1985.
94. Thalheim B. Dependencies in Relational Databases, 1991, Teubner-Texte zur Mathematik, 214 Pages
95. Chamberlin D.D. "Relational Data-Base Management Systems," Computing Surveys, Vol. 8, No. 1, p. 43-66, March 1976
96. Goldstein R.C., Strnad A.L. "The MACAIMS Data Management System," Proceedings of t he ACM-SIGFIDST Workshop on Data Description, Acess and Control, Nov. 1970. ACM, New York, 1970, pp. 201-229.
97. Notley M.G. "The Peterlee IS/1 system," IBM UK Scientific Centre Report UKSC-0018, March 1972.
98. Todd S.J.P. "Peterlee relational test vehicle PRTV, a technical overview," IBM Scientific Centre Report UKSC 0075, Peterlee, England, July 1975.
99. Whitney V.K.M. "RDMS: A Relational Data Management System," Proceedings of the Fourth International Svmposium on Computer and Information Sciences (COINS IV), Dec. 1972, Plenum Press, New York, 1972.
100. Pecherer R.M. " Efficient evaluation of expressions in a relational algebra," Proc. ACM Pacific 76 Regional Conf., April 1975, ACM, New York, 1975, pp. 44-49.
101. Gotlieb L.R. "Computing joins of relations, Proc. ACM-SIGMOD International Conference on Management of Data (San Jose, Calif., May 14-16, 1975), ACM. New York, 1975, pp. 55-63
102. Smith J.M., Chang P. "Optimizing the performance of a relational algebra data base interface," Comm. ACM 18, 10 (Oct. 1975), pp. 568-579.
103. Hall P. A. V. Optimisation of a single relational expression in a relational data base system, IBM Scientific Centre Renort UKSC 0076. Peterlee, England, July 1975.
104. Palermo F.P. An APL environment for testing relational operators and data base search algorithms. Proc. APL 75 Conf., June 1975, ACM. New York, 1975, pp. 249-256
105. Bui D.B., Skobelev V.G. Complexity of operations in database systems (a survey), Radioelectronic and computer systems, 2014, No 6(70). pp. 53-59
106. Held G.D., Stonebraker M.R., Wong E. "INGRES: a relational data base system," Proc. AFZPS h'ational Computer Conf., May 1975, Vol. 44, AFIPS Press, Montvale, N.J., 1975, pp 409-416.
107. Bracchi G., Fedeli A., Paolini P. A language for a relational data base management system. Proc. Sixth Annual Princeton Conf. on Information Science and Systems, March 1972, Princeton Univ., N.J., 1972. pp. 84-92.
108. Fehder P.L. The representation-independent language. Res. Rep. RJ 1121, IBM Research Laboratory, San Jose, Calif., Nov. 1972
109. McDonald N.,Stonebraker M. "CUPID — The Friendly Query Language," University of California, Berkeley, Technical Report No. UCB/ERL M487, October 1974. http://www2.eecs.berkeley.edu/Pubs/TechRpts/1974/ERL-m-487.pdf
110. McDonald N.,Stonebraker M Cupid--The friendly query language. Proc. ACM- Pacific-75, San Francisco, Calif., April 1975, pp. 127-131.
111. McDonald N. "Cupid: A Graphics Oriented Facility for Support of Non-Programmer Interactions with a Data Base," University of California, Berkeley, Technical Report No. UCB/ERL M563, November 1975.
112. McDonald N., Stonebraker M. "CUPID: the friendly query language," Proc. ACM Pacific 75 Regional Conf., Auril 1975. ACM. New York. 1975, pp, 127-131.
113. Zloof M.M."Query by example " RC4917, IBM T. J. Watson Research Center. Yorktown Heights, N. Y., July 1974.
114. Zloof M.M. "Query by Example," Proc. AFIPS National Computer Conf., May 1975, Vol. 44, AFIPS Press, Montvale, N.J., 1975, pp 431-438.

115. Zloof M.M. “Query by Example: the invocation and definition of tables and forms,” Proc. Internatl. Conf. on Very Large Data Bases, Sept. 1975, ACM, New York, 1975, pp. 1-24.
116. Zloof M.M. Query-by-Example: a data base language. IBM System J., 16:4, 1977, pp. 324–343
117. Thomas J. C., Gould J.D. “A psychological study of Query by Example,” Proc. AFIPS National Computer Conf., May 1975, Vol. 44, AFIPS Press, Montvale, N.J., p 439-445.
118. Zloof M.M. Office-by-Example: A business language that unifies data and word pro-cessing and electronic mail. IBM Systems Journal (Volume: 21, Issue: 3, 1982). Page(s): 272 - 304
119. Boyce R.F., Chamberlin D.D., King W.F., Hammer M.M. Specifying queries as relational expressions. Proc. ACM SIGPLAN/ SIGIR Interface Meeting, Gaithersburg, Md., Nov. 1973.
120. Boyce R.F., Chamberlin D.D., King W.F., Hammer M.M. Specifying queries as relational expressions: the SQUARE data sublanguage. Communications of the ACM, 1975, Volume 18, No 11, p. 621-628
121. Boyce, R.F., Chamberlin D.D. Using a structured English query language as a data definition facility. Res. Report RJ 1318, IBM Res. Lab., San Jose, Calif., Dec. 1973.
122. Chamberlin D D., Boyce R.F. SEQUEL: A structured English query language. SIGFIDET ‘74: Proceedings of the 1974 ACM SIGFIDET (now SIGMOD. workshop on Data description, access and control. May 1974 Pages 249–264.
123. Astrahan M.M., Chamberlin D.D. Implementation of a structured English query language. Communications of the ACM, 1975, Volume 18, No 10 pp. 580-588
124. Lorie R.A. XRM-an extended (n-ary. relational memory. Tech. Report G320-2096, IBM Scientific Center, Cambridge, Mass., Jan. 1974.
125. Astrahan M.M., Lorie R.A. “SEQUEL-XRM: a relational system,” Proc. ACM Pacific 76 Regional Conf., April 1975, ACM, New York, 1975, pp. 34-38.
126. Symonds A.J., Lorie, R. A. “A schema for describing a relational data base,” Proc. ACM-SIGFIDET Workshop on Data Description and Access, Nov. 1970, ACM, New York, 1970, pp. 230-245.
127. Lorie R.A., Symonds,A.J. “A relational access method for interactive applications,” Courant Computer Science Symposia, 6, Data Base Systems, Prentice-Hall, New York, 1971, pp 99-124.
128. Chamberlin D D., Astrahan M.M., Eswaran K.P., Griffiths P.P., Lorie R.A., Mehl J.W., Reisner Ph., Wade B.W. SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control. IBM Journal of Research and Development 20(6): 560-575 (1976)
129. Carlson C.R., Kaplan R.S. A Generalized Access Path Model and Its Application to a Relational Data Base System. SIGMOD ‘76: Proceedings of the 1976 ACM SIGMOD international conference on Management of data. June 1976, Pages 143–154
130. Notley M, “Peterlee IS/1 System”, UKSC Report 18, 1972
131. Todd S. “The Peterlee Relational Test Vehicle - A System Overview”. IBM Systems Journal. 1976, 15 (4): 285–308.
132. Astrahan M.M., et al. System R: A relational approach to database management. ACM Trans. Database Syst. Vol. 1, No 2 (June 1976), 97-137
133. Chamberlin D.D. A summary of user experience with the SQL data sublanguage. Proc. Internat. Conf. Data Bases, Aberdeen, Scotland, July 1980, pp. 181-203
134. Chamberlin D.D., et al. Support for repetitive transactions and ad-hoc queries in System R. ACM Trans. Database Syst. Vol. 6, No 1 (March 1981), 70-94.
135. Chamberlin D.D., Gilbert, A.M., Yost, R.A. A history of System R and SQL/data system. VLDB ‘81: Proceedings of the seventh international conference on Very Large Data Bases - Volume 7, September 1981, pp. 456–464
136. McDonald N., Stonebraker M., Wong E. “Preliminary design of INGRES: Part I,” Electronics Research Lab. Report ERL-M435, Univ. of California, Berkeley, April 1974. http://www2.eecs.berkeley.edu/Pubs/TechRpts/1974/ERL-m-435.pdf
137. McDonald N., Stonebraker M., Wong E. “Preliminary design of INGRES: Part II,” Electronics Research Lab. Report ERL-

M436, Univ. of California, Berkeley, April 1974. http://www2.eecs.berkeley.edu/Pubs/TechRpts/1974/ERL-m-436.pdf

138. Held G., Stonebraker M. "Storage structures and access methods in the raational data base management system INGRES," Proc. ACM Pacific 75 Regional Conf., April 1975, ACM, New York, 1975, pp 26-33.
139. Wong E., Youssefi K. Decomposition--A strategy for query processing. ACM Trans. on Database Systems I, 3 (Sept. 1976), 223-241
140. Stonebraker M. "Implementation of integrity constraints and views by query modification," Proc. ACM-SIGMOD Conf. May 1975, ACM, New York, 1975, pp 65-78.
141. Stonebraker M., Wong E. Access control in a relational data base management system by query modification. Proc. 1974 ACM Nat. Conf., San Diego, Calif., Nov. 1974, pp. 180-187.
142. Stonebraker M. "High level integrity assurance in relational data base management systems," Electronics Research Lab. Report ERL-M473, Univ. of Calif. at Berkeley, August 1974.
143. Stonebraker M., Rubinstein P. The INGRES protection system. Proc. 1976 ACM National Conf., Houston, Tex., Oct. 1976
144. Stonebraker M., Held G., Wong E., Kreps P. "The Design and Implementation of INGRES". ACM Transactions on Database Systems. Vol.1, No 3. 1976 pp.189–222.
145. Stonebraker M., Rowe L. "The design of POSTGRES," in Proc. 1986 ACM-SIGMOD Conf., Washington, DC, June 1986.
146. Rowe L.A., Stonebraker M. "The POSTGRES data model," in Proc. 13th Intl. Conf. on Very Large Data Bases, P. M. Stocker, W. Kent, P. Hammersley, Eds., San Francisco, CA: Morgan Kaufmann Publishers Inc., 1987, pp. 83-96.
147. Stonebraker M. " The design of the POSTGRES storage system ", in Proc. 1987 VLDB Conf:, Brighton, England, Sept. 1987.
148. Stonebraker M., Hanson E., Hong C. H. "The design of the POSTGRES rules system", Proc. IEEE Conference on Data Engineering, Feb. 1987.
149. Stonebraker M., Rowe L.A., Hirohama M. The Implementation Of Postgres IEEE Transactions on Knowledge and Data Engineering, 1990, Vol. 2, No 1, pp. 125 - 142
150. ACM Turing Award Goes to Pioneer in Database Systems Architecture: MIT's Michael Stonebraker Brought Relational Database Systems from Concept to Commercial Success. - https://www.prweb.com/pdfdownload/12607207.pdf
151. Kuznetsov S.D. Methods for optimization of query execution in relational DBMS (Rus) // "Vychislitelnye nauki. Vol. 1 (Itogi nauki i tekhniki VINITI AN USSR" M.; VINITI AN USSR, 1989.- 76-153. - http://masters.donntu.org/2002/foreign/aswad/lib/mpbd.htm или http://citforum.ru/database/articles/art_26.shtml
152. Brodie M.L., Schmidt J.W. Final Report of the ANSI/X3/SPARC DBS-SG Relational Database Task Group. SIGMOD Record 12(4): i-62 (1982).

Литература по транзакциям

153. Gray J. The Transaction Concept: Virtues and Limitations. In: Proceedings of the 7th International Conference on Very Large Databases, 1981. pp. 144—154, IEEE, Cannes, France,
154. Advanced Transaction Models and Architectures. Sushil Jajodia and Larry Kerschberg (eds.) Springer Scienee+Business Media New York. 1997
155. Gray J., Lorie R., Putzulo G. "Granularity of Locks and Degrees of Consistency in a Shared Data Base," In Modelling in Data Base Management Systems. G.M. Ni]ssen, (ed.) North Holland Publishing Company, 1976, pp.365-394
156. Reis D.R., Stonebraker M. Effect of locking granularity in a database management systems. ACM Trans. on Database Syst., 2:3, 1977, pp. 233-246.
157. Gray J.N. Notes on data base operating systems. In: Bayer R., Graham R.M., Seegmüller G. (eds) Operating Systems. Lecture Notes in Computer Science, vol 60. Springer, Berlin, Heidelberg. 1978. p. 393–481.
158. Eswaran K.P, Gray J, Lorie R.A, Traiger I.L. The notions of consistency and predicate locks in a database system. Commun ACM. 1976;19(11):624–633.
159. Lampson B.W. Atomic transactions. In: Lampson B.W, Paul M, Siegert H.J, editors. Distributed systems – architecture and

implementation: an advanced course, LNCS, vol. 105. Berlin: Springer; 1981. p. 246–285.
160. Lomet D.B. Process structuring, synchronization, recovery using atomic actions. ACM SIGPLAN Not. 1977; 12(3):128–137.
161. Bernstein P.A, Shipman D.W, Wong W.S. Formal aspects of serializability in database concurrency control. IEEE Trans Software Eng. 1979;SE-5(3): 203–216.
162. Bernstein P.A, Hadzilacos V, Goodman N. Concurrency control and recovery in database systems. Reading: Addison-Wesley; 1987.
163. Papadimitriou C.H. The serializability of concurrent database updates. J ACM. 1979;26(4):631–653.
164. Papadimitriou C.H. The theory of database concurrency control. Rockville: Computer Science; 1986.
165. Weikum G, Vossen G. Transactional information systems – theory, algorithms, the practice of concurrency control and recovery. San Francisco: Morgan Kaufmann; 2002.
166. Shasha D, Bonnet P. Database tuning – principles, experiments, and troubleshooting techniques. San Francisco: Morgan Kaufmann; 2003.
167. Ramamritham, K., Chrysanthis, P. K., (1997). Advances in Concurrency Control and Transaction Processing. IEEE Computer Society Press, Los Alamitos, California.
168. Grefen P., Apers P. (1993). Integrity Control in Relational Database Systems - An Overview. Journal of Data & Knowledge Engineering (10)2: 187-223.
169. Moss J.E.B. Nested transactions: an approach to reliable distributed computing. Technical Report. PhD Thesis. UMI Order Number: TR-260: Massachusetts Institute of Technology; 1981. p. 178.
170. Been C, Bernstein P.A., Goodman N, Lai M.Y., Shasha D.E. A concurrency control theory for nested transactions. Proc. of Second ACM Symposium on Principles of Database Systems (PODS), 1983, pp. 45-62
171. Davies C.T. Data processing spheres of control. IBM Syst J. 1978;17(2):179–198.
172. Dayal U., Hsu M., Ladin R. A generalized transaction model for long-running activities and active databases. IEEE Data Engineering Bulletin, March 1991, vol. 14, No 1,. pp 4-8
173. Weikum G. and Schek H. Concepts and applications of multilevel transactions and open-nested transactions. In Elmagarmid A., editor. Database Transaction Models for Advanced Applications. Morgan Kaufmann Publishers. San Mateo. CA., 1992, pp. 515–553.
174. Weikum G. Principles and realization strategies of multilevel transaction management. ACM Transactions on Database Systems. 1991;16(1):132–180.
175. Krychniak P., Rusinkiewicz M., Chichocki A., Sheth A., Thomas G. Bounding the Efects of Compensation under Relaxed Multi-Level Serializability. Distributed and Parallel Database Systems, 1996, 4(4), pp. 355-374
176. Lewis, P. M., Bernstein A. J., Kifer M. (2002). Databases and Transaction Processing: An Application-Oriented Approach. Addison-Wesley, United States
177. Breitbart Y., Garcia-Molina H., Silberschatz A. Overview of multidatabase transaction management. VLDB Journal, 1992, vol. 1, No 2, pp. 181-240.
178. X/Open Company Ltd., (1996). Distributed Transaction Processing: Reference Model, version 3. X/Open Company Ltd., U.K.
179. Elmagarmid A.K., Leu Y., Litwin W.,Rusinkiewicz M. (1990) A Multidatabase Transaction Model for InterBase. In Proc. of the 16th. Intl. Conference on Very Large Data Bases, pp. 507-518, Brisbane. Australia
180. Zhang A., Nodine M., Bhargava B., Bukhres O. Ensuring Relaxed Atomicity for Flexible Transactions in Multidatabase Systems. In Proc/ 1994 SIGMOD International Conference on Management of Data, 1994, pp. 67-78
181. Zhang A, Nodine M, Bhargava B. Global scheduling for flexible transactions in heterogeneous distributed database systems. IEEE Trans Knowl Data Eng. 2001;13(3):439–450.
182. Wächter H, Reuter A. The ConTract model. In: Elmagarmid A.K., editor. Database transaction models for advanced applications. Los Altos: Morgan Kaufmann; 1992. pp 39-43

183. Veijalainen J., Eliassen F. The S—transaction Model. In: Elmagarmid A.K., editor. Database transaction models for advanced applications. Los Altos: Morgan Kaufmann, 1992, pp. 55-59
184. Chen J., Bukhres O., Elmagarmid A. K. (1993). IPL: A Multidatabase Transaction Specification Language. In Proc. of the 13th Intl. Conference on Distributed Computing Systems - ICDCS '93. 1993, pp. 439-448
185. Garcia-Molina H. Using Semantic Knowledge for Transaction Processing in a Distributed Database. ACM Transactions on Database Systems, 8(2):186-213, June 1983.
186. Korth H., Levy E., Silberschatz A. A Formal Approach to Recovery by Compensating Transactions. In Proceedings of the 16th International Conference on Very Large Data Bases, Brisbane, Australia, 1990, pp. 95–106
187. Garcia-Molina H., Salem K. Sagas. In Proc. of ACM SIGMOD International Conference on Management of Data, 1987, pp 249-259 San Francisco, CA.
188. Bancilhon F., Kim W., Korth H. A model of CAD Transactions. VLDB '85: Proceedings of the 11th international conference on Very Large Data Bases - Volume 11, 1985, pp. 25–33
189. Garcia-Molina. H., Salem K., Gawlick D., Klein J., Kleissner K., Modeling Long-Running Activities as Nested Sagas, IEEE Data Engineering Bulletin, 1991, 14(1) pp 14-18
190. Pu C., Kaiser G.E., Hutchinson N.C. Split-transactions for open-ended activities. In: Proceedings of the 14th International Conference on Very Large Data Bases; 1988. p. 26–37.
191. Kaiser G.E., Pu C. Dynamic restructuring of transactions. In: Elmagarmid AK, editor. Database transaction models for advanced applications. Burlington: Morgan Kaufmann Publishers; 1992. p. 265–295.
192. Chrysanthis P.K, Ramamritham K. Synthesis of extended transaction models using ACTA. ACM Trans. Database Syst. 1994;19(3):450–491.
193. Nodine M.H., Zdonik S.B. Cooperative transaction hierarchies: Transaction support for design applications. VLDB Journal, 1(1):41–80, 1992.
194. Chrysanthis P.K., Ramamritham, K., (1990). ACTA: A Framework for Specifying and Reasoning about Transaction Structure and Behavior. Proceedings of the ACM SIGMOD International Conference on Management of Data: 194-203.
195. Chrysanthis P.K., Ramamritham K. (1992). ACTA: The SAGA Continues. In Elmagarmid A., editor. Database Transaction Models for Advanced Applications. Morgan Kaufmann Publishers. San Mateo. CA., 1992, pp. 349-397
196. Biliris A., Dar S., Gehani N., Jagadish H., Ramamritham K. (1994). ASSET: A System for Supporting Extended Transactions. In Proc. of ACM SIGMOD Conference on Management of Data, pages 44-54, Minneapolis, M.N.
197. Abbott R., Garsia-Molina H. Scheduling real-time transactions: a performance evaluation. ACM Trans, on Database Syst, 17(3), September 1992, pp. 513-560
198. Agrawal D., El Abbadi A., Jeffers R. Using Delayed Commitment in Locking Protocols for Real-Time Databases. SIGMOD Conference 1992: 104-113
199. Hong D., Johnson T., Chakravarthy S. Real-Time Transaction Scheduling: A Cost Conscious Approach. SIGMOD Conference 1993: 197-206.
200. Alonso R., Korth H. Database System Issues in Nomadic Computing. SIGMOD Record, Vol. 22, No 2,1993, pp.388-392.
201. Imelinski T., Badrinath B.R. Data Management for Mobile Computing. SIGMOD Record, Vol. 22, No. 1, 1993
202. Ceponkus A., Dalal S., Fletcher T., Furniss P., Green A., Pope B. Business transaction protocol, Version 1.1, 2002
203. Business transaction protocol. - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=businesstransaction [2004]
204. Stevens M., Mathew S., McGovern J., Tyagi S. Java Web Services Architecture. San Francisco: Morgan Kaufmann Publishers, 2003.
205. WSTx (Web Services Transactions). - https://searchapparchitecture.techtarget.com/definition/WSTx-Web-Services-Transactions
206. IBM, BEA Systems, Microsoft, Arjuna, Hitachi, IONA, \Web Services Transactions

specifations," IBM Developer Works, IBM, 2004.
207. Curbera F., Khalaf R., Mukhi N., Tai S.,Weerawarana S. The Next Step in Web Services," Communications of the ACM, October 2003, Vol. 46, No. 10, Pages 29-34
208. OASIS Web Services Composite Application Framework (WS-CAF), OASIS, 2006. - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-caf
209. Little M., Freund Th. J.. A comparison of web services transaction protocols: A comparative analysis of WS-C/WS-Tx and OASIS BTP," IBM, 2003. Available: http://www-128.ibm.com/developerworks/webservices/library/ws-comproto/. [Accessed May 2008].
210. Kratz B., Protocols For Long Running Business Transactions. Technical Report 17, Infolab Technical Report Series, 2004, 48 p.
211. Jin T., Goschnick S. (2004) Utilizing Web Services in an Agent Based Transaction Model. In: Cavedon L., Maamar Z., Martin D., Benatallah B. (eds) Extending Web Services Technologies. Multiagent Systems, Artificial Societies, and Simulated Organizations (International Book Series), vol 13. Springer, Boston, MA. pp 273-291

Одержано 24.06.2021

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

ПРОБЛЕМИ ПРОГРАМУВАННЯ.
2021 – № 3

УДК 004.272.26 + 004.021 + 004.032.26

Розподілена реалізація методу нейроеволюції наростаючої топології / І. З. Ашур, А. Ю. Дорошенко.

Попри сильні сторони методу нейроеволюції наростаючої топології, як-от можливість його застосування в завданнях, де важко обрати функцію витрат та топологію нейронної мережі, однією з проблем нейроеволюції та методу нейроеволюції наростаючої топології є повільна конвергенція до оптимальних результатів, особливо у випадку роботи з комплексними та складними середовищами. У роботі запропонована нова розподілена реалізація методу нейроеволюції наростаючої топології, яка, за наявності достатніх обчислювальних ресурсів, дозволяє радикально збільшити швидкість знаходження оптимальних конфігурацій нейронних мереж. З метою оптимізації продуктивності рішення, рівномірного розподілу завдань між вузлами та оптимального використання обчислювальних ресурсів була реалізована підтримка пакетної оцінки геномів. Експериментальна перевірка нової реалізації засвідчує, що використовуючи запропоноване розподілене рішення, швидкість виконання методу нейроеволюції наростаючої топології в частині оцінювання згенерованих нейронних мереж на прикладі розглянутого завдання і середовища може зростати на декілька порядків.

Ключові слова: NEAT, нейроеволюція наростаючої топології, штучні нейронні мережі, навчання з підкріпленням, генетичні алгоритми, розподілені обчислення, хмарні обчислення.

UDC 004.272.26 + 004.021 + 004.032.26

Distributed implementation of neuroevolution of augmenting topologies method / Achour I., Doroshenko.

Despite the neuroevolution of augmenting topologies method strengths, like the capability to be used in cases where the formula for a cost function and the topology of the neural network are difficult to determine, one of the main problems of such methods is slow convergence towards optimal results, especially in cases with complex and challenging environments. This paper proposes the novel distributed implementation of neuroevolution of augmenting topologies method, which considering availability of sufficient computational resources allows drastically speed up the process of optimal neural network configuration search. Batch genome evaluation was implemented for the means of proposed solution performance optimization, fair, and even computational resources usage. The proposed distributed implementation benchmarking shows that the generated neural networks evaluation process gives a manifold increase of efficiency on the demonstrated task and computational environment.

Key words: NEAT, neuroevolution of augmenting topologies, artificial neural networks, reinforcement learning, genetic algorithms, distributed computing, cloud computing.

УДК 004.94

Визначення ступеня семантичної подібності з використанням апарату дескриптивних логік / О. В. Захарова.

Встановлення семантичної подібності інформації є невід'ємою складовою процесу вирішення будь –яких задач інформаційного пошуку, в тому числі задач, пов'язаних з обробкою великих даних, виявленням семантичних веб сервісів, категоризації та класифікації інформації тощо. Введення спеціальних функцій для визначення кількісних показників ступеня семантичної відповідності інформації дозволяють ранжувати знайдену інформацію за її семантичною близькостю до цілі або пошукового запиту/шаблону. Формування таких оцінок повинно враховувати багато аспектів від сутності самих оцінюваних понять, до особливостей бізнес-задачі, в межах вирішення якої це робиться. Зазвичай, при побудові функцій подібності семантичні підходи поєднуються зі структурними, що забезпечують синтаксичне порівняння описів концептів. Це дозволяє деталізувати опис концепта, а вплив синтаксичної відповідності можна значно зменшити, використовуючи для представлення інформації виразніші дескриптивні логіки (ДЛ) та шляхом перенесення фокусу на семантичні властивості. ДЛ-онтології на сьогодні є найбільш розвинутим засобом представлення семантики, а механізми міркувань ДЛ забезпечують можливість логічного висновку. Більшість наведених у роботі оцінок будуються на основі базових ДЛ, що підтримують лише конструктор перетину, але описані підходи можуть бути застосовані для будь-якої ДЛ, що забезпечує базові сервіси міркувань.

У роботі проведений аналіз існуючих підходів, моделей та мір оцінювання, що засновані на застосуванні апарату ДЛ, запропонована їхня класифікація як за рівнем визначення подібності, так й за видами співставлення. Головна увага приділяється встановленню подібності концептів. Задачі встановлення подібності між екземплярами/концептом та екземпляром зводяться до знаходження найбільш специфічного концепту для екземпляра/екземплярів та оцінювання подібності відповідних концептів. Введено поняття екзистенційної подібності та продемонстровано застосування певних видів оцінок для визначення ступеня подібності понять/знань на прикладі онтології геометричних понять.

UDC 004.94

Defining degree of semantic similarity using description logic tools / O. Zakharova.

Establishing the semantic similarity of information is an integral part of the process of solving any information retrieval tasks, including tasks related to big data processing, discovery of semantic web services, categorization and classification of information, etc. The special functions to determine quantitative indicators of degree of semantic similarity of the information allow ranking the found information on its semantic proximity to the purpose or search request/template. Forming such measures should take into account many aspects from the meanings of the matched concepts to the specifics of the business-task in which it is done. Usually, to construct such similarity functions, semantic approaches are combined with structural ones, which provide syntactic comparison of concepts descriptions. This allows to do descriptions of the concepts more detail, and the impact of syntactic matching can be significantly reduced by using more expressive descriptive logics to represent information and by moving the focus to semantic properties. Today, DL-ontologies are the most developed tools for representing semantics, and the mechanisms of reasoning of descriptive logics (DL) provide the possibility of logical inference. Most of the estimates presented in this paper are based on basic DLs that support only the intersection constructor, but the described approaches can be applied to any DL that provides basic reasoning services.

This article contains the analysis of existing approaches, models and measures based on descriptive logics. Classification of the estimation methods both on the levels of defining similarity and the matching types is proposed. The main attention is paid to establishing the similarity between concepts (conceptual level models). The task of establishing the value of similarity between instances and between concept and instance consists of finding the most specific concept for the instance / instances and evaluating the similarity between the concepts. The term of existential similarity is introduced. In this paper the examples of applying certain types of measures to evaluate the degree of semantic similarity of notions and/or knowledge based on the geometry ontology is demonstrated.

Ключові слова: семантична подібність інформації, найменше спільне покриття, оцінки вимірювання подібності, найбільш специфічний концепт, найбільш специфічний попередник, функція подібності, подібність за інформаційним змістом, семантична подібність за відповідністю ознак, функція відстані шляху, моделі оцінювання на основі властивостей, моделі оцінювання на основі семантичної мережі, моделі оцінювання на основі інформаційного контенту, екзистенційна подібність концептів, подібність екземплярів, подібність концепту та екземпляра, подібність ДЛ описів, GCS-подібність.

Keywords: semantic similarity of information, a value of similarity of concepts, least concept subsumer, measures for similarity evaluating, most specific concept, most specific is-a ancestor, similarity function, similarity measure information content, features-based similarity measure, measure of distance between concepts, features-based models, semantic-network based models, information content based models, existential concepts similarity, similarity between two individuals, similarity between concept and individual, similarity between DL-descriptions of concepts, GCS-similarity.

УДК 004.75

На шляху до створення української національної хмари відкритої науки / С.Я. Свістунов, П.І. Перконос, С.В. Суботін, Є.М. Твердохліб.

Сучасні умови розвитку науки, які характеризуються значним зростанням об'ємів інформації, вимагають нових підходів до обчислювальних способів і засобів її обробки. В статті розглянуті підходи до вирішення зазначених викликів часу, а саме - створення на базі використання хмарних технологій умов і структур для більш ефективної співпраці наукових колективів, що працюють над схожими науковими проблемами.
Проаналізовано досвід побудови та розвитку European Open Science Cloud. Викладена стратегія побудови Національної хмари відкритої науки в Україні. Наведено основні результати створення в Національній Академії наук України спільного інформаційного середовища для наукових досліджень як прототипу Національної хмари відкритої науки інтегрованої до Європейської хмари відкритої науки.

Ключові слова: інформатизація наукових досліджень, національна хмара відкритої науки, хмарна інфраструктура, Європейська хмара відкритої науки

UDC 004.75

On the way to creating Ukrainian national cloud of open science / Svistunov S., Perkonos P., Subotin S., Tverdochlib Ya.

Modern conditions of science development, which are characterized by a significant increase in the volume of information, require new approaches to computational methods and new approaches to information processing. The article considers approaches to creating conditions and tools based on cloud technologies for more effective cooperation of research teams working on similar scientific problems.
The article analyzes the stages of development of the European Open Science Cloud and presents the strategy of building the National Open Science Cloud. The article presents the main results of the development of a common information resource for scientific researches at the National Academy of Sciences of Ukraine, which can be considered as a prototype of the National Open Science Cloud, which integrates into the European Open Science Cloud.

Key words: informatization of scientific research, national cloud of open science, cloud infrastructure, European Open Science Cloud.

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 стандар¬тизація, управління транзакціями. Етап розширених реляційних баз даних присвячений опису темпоральних, просторових, дедуктивних, активних, об'єктних, розподілених та статистичних баз даних, баз даних масивів, 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, 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, мультімодельна, хмарна, наукова, багатозначна, XML, NewSQL, онтологічна, великі дані.

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.

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: +380 (96) 418 3082.

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 – англійською і українською 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**.

Індекс 90853