Міністерство освіти і науки України

Національний університет «Львівська політехніка»

На правах рукопису

Болюбаш Юрій Ярославович

УДК 004.9:371.261

Методи та засоби опрацювання інформаційних ресурсів Великих даних в системах територіального управління

Спеціальність 01.05.03 – математичне та програмне забезпечення обчислювальних машин і систем

Дисертація на здобуття наукового ступеня
кандидата технічних наук

Науковий керівник:
Шаховська Н.Б.,
д.т.н., професор кафедри ІСМ
Національного університету
«Львівська політехніка»

Львів 2017

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

OEM (Object Exchange Model) – модель обміну об’єктами.

XML (Extensible Markup Language) – розширювана мова розмітки.

RDF (Resource Description Framework) – фреймворк опису ресурсів.

СЕП – соціально-економічні процеси.

CAP (Consistence, Availability, Partition tolerance) – узгодженість, доступність, стійкість до поділу.

ДВО – деструктивна відносна оцінка.

NoSQL – не тільки SQL.

OLAP (On-line Analitical Processing) – аналітична обробка в реальному часі.

SQL (Structuted Query Language) – структурована мова запитів.

СУБД – система управління базами даних.

ACID (*Atomicity, Consistency, Isolation, Durability*) – це набір властивостей, що гарантують надійну роботу транзакцій бази даних: атомарність, узгодженість, ізольованість, довговічність.

БД – база даних.

MOBIE (Mosaic Based Information Explorer) – провідник по розрізненим даним.

SGML (Standard Generalized Markup Language) – стандартна узагальнена мова розмітки.

DTD (Document Type Definition) – визначення типу документа.

Mathml (Mathematics Markup Language) – мова математичної розмітки.

SMIL (*Synchronized Multimedia Integration Language*) – мова інтеграції синхронізованих джерел мультимедійної інформації.

CML (*Chemistry Markup Language*) – мова хімічної розмітки.

XSD (XML Schema Definition) – визначення схеми XML.

XSLT(extensible Stylesheet Language Transformations) – мова перетворення XML-документів.

Xpath (XML Path)– мова виразів, для визначення частини XML документа, або для обчислення величин (наприклад, рядкових, числових або булевих) на основі вмісту XML документа.

Xlink (XML Linking) – мова розмітки, що дозволяє вставляти в XML документи елементи, щоб створити і описати посилання між ресурсами.

Xpointer (XML Pointer) – розширювана специфікація, що визначає способи адресації структурних елементів і фрагментів документів у форматі XML.

*URI (*Universal R*esource Identifier)* – уніфікований ідентифікатор ресурсів.

ПППД – платформа підтримання простору даних.

РБД – реляційна база даних.

*MVVM («*View-Model*»)* – «Вигляд-Модель».

*MCV (*Model-View-Controller) – Модель-Вигляд-Контролер.

МНК – метод найменших квадратів.

МЛР – множинна лінійна регресія.

МГВА – метод групового врахування аргументів.

ПЗ – програмне забезпечення.

ВСТУП

Застосування систем територіального розвитку сприяє швидкому поширенню знань, навичок та найкращих практик у певних географічних межах, таких як місто, регіон, країна тощо. Для комплексного аналізу інформації на рівні регіону необхідно:

- зберігати і керувати інформацією розміром у петабайти;
- опрацьовувати інформацію з реляційних, багатовимірних баз даних, баз даних XML і NoSQL, структурованих і слабоструктурованих текстових файлів, баз геоданих, медіафайлів тощо;
- аналізувати різнотипну інформацію, використовуючи як консолідований, так і федеративний підхід для її отримання.

Процес побудови узагальненої (комплексної) моделі регіону ускладнюється різноманітністю моделей даних, а також через наявність різних рівнів агрегації даних. Однією з популярних технологій для розроблення систем територіального управління є Великі дані.

Великі дані є терміном, який використовується для ідентифікації наборів даних, з якими не можна впоратися з використанням існуючих методологій та програмних засобів через їх великий розмір і складність. Такі дослідники як М. Гілбернт, С. Стрініваса та ін. розробили методики і програмні засоби для передачі даних або видобування інформаційних гранул з Великих даних (колекції об’єктів, які зазвичай формуються для атрибутів з числовими даними і які розташовані поряд через їх схожість, функціональну або фізичну суміжність). Методи машинного навчання та візуалізації даних дають змогу опрацювати та графічно подати результати аналізу даних великих обсягів (мільойони кортежів). Проте нерозв’язаною задачею залишається задача побудови відображення між моделями даних різних джерел. В роботах Alejandro Maté [140] та Lucentia Research Group обґрунтовано використання

багатовимірної моделі для представлення Великих даних та побудови відображення в реляційну модель. Проте у випадку використання бази даних ключ-значення як однієї з джерел даних застосування багатовимірної моделі неприйнятне. Vinayak Borkar [131], Yingyi Bu [132] пропонують використовувати об'єктно-орієнтований підхід, проте обмеженням є кількість зв'язків між об'єктами.

Отже, єдиного підходу до опрацювання Великих даних не знайдено.

Коли аналізувати можливість використання Великих даних в системах територіального управління, то маємо:

- великий набір сутностей: особи, місця, організації (фізичні, юридичні), дати, природні ресурси (річки, ліси, озера), рекреаційний фонд (історичні пам'ятки, санаторії), законодавчі акти та звіти;
- база даних особливостей: документи для інтелектуального аналізу даних, онтологічні терміни, словники даних, які дозволяють зв'язати деякі об'єкти.

Грунтуючись на цій інформації, з метою її подальшого аналізу, слід вирішити питання, які сутності і в який спосіб пов'язані між собою.

Тому задача розроблення методів та засобів опрацювання Великих даних в системах регіонального розвитку є актуальною.

Зв'язок роботи з науковими програмами, планами, темами

Дисертаційна робота виконувалась в рамках пріоритетного наукового напрямку, затвердженого в числі актуальних проблем Міністерством освіти і науки України, зокрема у рамках науково-дослідної теми «Комплекс інтелектуальних інформаційних технологій інтеграції даних для обліку та аналізу підвищення кваліфікації вчителів», номер держреєстрації 0113U005273.

Автор розробив модель даних «сутність-характеристика» для подання та аналізу різнорідних даних.

Мета і задачі дослідження

Метою дисертаційної роботи є розробка методів та засобів опрацювання та аналізу різнорідної інформації на основі Великих даних для процесів територіального управління.

Мета дисертаційної роботи визначає необхідність розв'язання таких задач:

1) проаналізувати методи, моделі та засоби опрацювання різнотипних даних, інформаційні технології роботи з Великими даними, обґрунтування й постановка задачі;
2) розробити інформаційну модель Великих даних «сутність-характеристика»;
3) розробити метод перетворення реляційних та слабоструктурованих даних у дані, подані в моделі «сутність-характеристика»;
4) розробити метод формування відповіді на запит користувача до Великих даних;
5) розробити програмні компоненти інформаційної системи підтримки прийняття рішень для управління регіоном з використанням Великих даних.

Об'єктом дослідження є процес інтеграції слабоструктурованих даних, виділення елементів структури, та збереження їх у базу даних.

Предметом дослідження є методи та засоби опрацювання Великих даних.

Методи дослідження. Для досягнення поставленої мети використовувались: методи системного аналізу – для формування концептуальної моделі Великих даних; методи штучного інтелекту – для виявлення закономірностей у каталозі Великих даних; методи об'єктно-орієнтованого аналізу та проектування – для визначення семантичних зв'язків між джерелами даних.

Наукова новизна одержаних результатів

Наукова новизна роботи полягає у розв'язанні актуального наукового завдання розроблення методів та засобів організації та інтеграції інформаційних ресурсів Великих даних. Отримано такі нові наукові результати:

- *вперше:* розроблено модель Великих даних «сутність-характеристика», яка дає змогу опрацьовувати структуровані та напівструктуровані дані та на відміну від багатовимірної моделі не містить надлишковості;
- *удосконалено:* отримання відповіді на запит користувача шляхом уніфікації елементів мов запитів до різних інформаційних джерел, що дало змогу трансформувати запит до Великих даних у запит на основі ключових слів;
- *одержав подальший розвиток:* федеративний метод інтеграції даних за рахунок визначення пари «сутність-характеристика» та узгодження семантики, що на відміну від методів інтеграції даних на рівні сховища даних дозволило інтегрувати дані з джерел з наперед невідомою структурою даних без попереднього локального завантаження і що, в свою чергу, дало змогу підвищити ефективність подальшого аналізу Великих даних.

Практичне значення одержаних результатів

Практичне значення отриманих результатів полягає у тому, що:

- удосконалено алгоритми інтеграції інформаційних ресурсів за допомогою попереднього визначення структури джерел даних та їх узгодження, що дало можливість розробити алгоритм оцінювання якості Великих даних;
- розроблено алгоритми пошуку даних за запитом користувача з метою уніфікації алгоритмів опрацювання даних з різнотипних джерел даних, що дозволило збільшити релевантність відповіді користувачеві на 2%;

- спроектовано архітектуру інформаційної системи для роботи з Великими даними.

Одержані у роботі результати використано під час розроблення програмних компонент інформаційної системи для збору та опрацювання даних «Інтегратор», впровадженої в Золочівській районній раді та Регіональній агломерації «Дрогобиччина». Розроблення впроваджені також в навчальному процесі при викладанні курсу «Бази даних» Золочівського коледжу НУ «Львівська політехніка».

Особистий внесок здобувача

Наукові результати, подані у дисертації, одержані здобувачем особисто. У друкованих працях, опублікованих у співавторстві, внесок здобувача такий: [3, 4, 5, 14] – визначено поняття Великих даних та базові характеристики; [9, 7, 11, 13] – розроблено модель «сутність-характеристика» та визначено асоціації з неструктурованими моделями даних; [6, 10, 15] – розроблено метод перетворення даних в модель «сутність-характеристика»; [8, 12] – спроектовано архітектуру Великих даних.

Апробація результатів дисертації

Основні результати дисертаційної роботи доповідалися на семінарах та конференціях: Міжнародних конференціях «The experience of designing and application of CAD systems in microelectronics» – CADSM (Львів-Поляна, 2015); X Міжнародна конференція «Комп'ютерні науки та інформаційні технології» – CSIT (Львів, 2015); Міжнародній конференції «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту» – ISDMCI'2013 (Євпаторія, 2013); II та III Міжнародних науково-практичних конференціях «Математика. Інформаційні технології. Освіта» (Луцьк-Світязь, 2013, 2014), III Міжнародній науково-практичній конференції «Інформаційні управляючі системи та технології» – ІУСТ (Одеса, 2014).

Публікації

Основні результати роботи відображені у 15 опублікованих працях, у тому числі 6 статей у наукових фахових виданнях України, 3 – у наукових періодичних виданнях іноземних держав, що входять до наукометричних баз даних, 6 – в збірниках праць конференцій.

РОЗДІЛ 1.
АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

У розділі подано регіон як складну систему. Визначено задачі, які виникають під час опрацювання різнотипних даних. Розглянуто особливості аналізу соціо-еколого-економічних даних, обгрунтовано доцільність їх опрацювання у формі Великих даних. Подано визначення Великих даних та описано їх основні характеристики. Проаналізовано математичні засоби подання та опрацювання Великих даних та визначено їх обмеження, а також програмні засоби роботи з Великими даними. Здійснено постановку задачі.

1.1. Аналіз регіону як складної системи

Регіон – складна, багатокомпонентна, відкрита, динамічна, ймовірнісна система. На підтвердження цього маємо:

- у склад регіону входять різні за природою об’єкти, такі як економічні показники, соціальні фактори, технічні об’єкти, які у свою чергу також є системою;
- стан системи змінюється під дією зовнішніх факторів;
- перехід з одного стану в інший відбувається не миттєво, а з часом;
- причинно-наслідкові зв’язки є ймовірнісного характеру.

Регіон – це цілісна територіальна частина господарства країни, яка характеризується наявністю місцевих органів управління, певною структурою виробництва, внутрішніми зв’язками, населенням, виробничою та соціальною інфраструктурою [134].

Характеристики регіону (рис. 1.1):

- великий набір сутностей: особи, організації, природні ресурси, рекреаційний фонд, законодавчі акти та звіти;

- база даних особливостей: документи для інтелектуального аналізу даних, словники даних для зв’язування об'єктів.

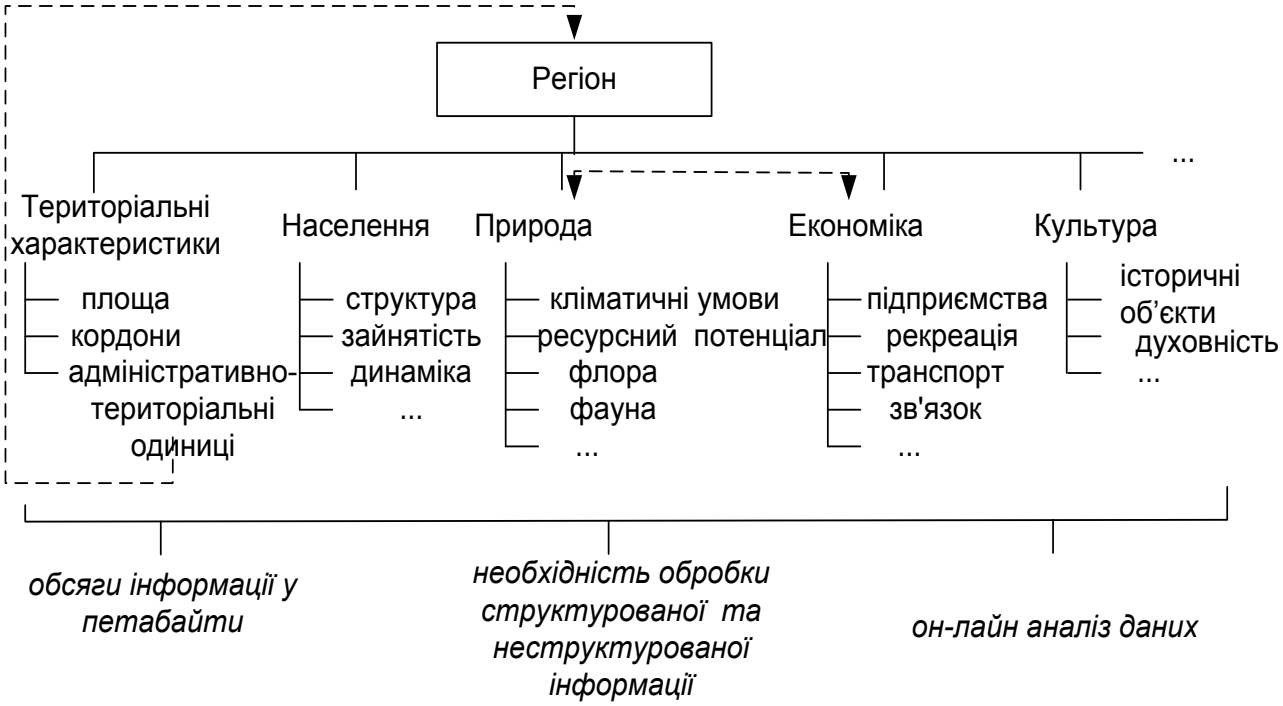


Рис. 1.1. Структура регіону

Відповідно до 3-х рівневої структури системи управління регіоном [1, 23] виділяють три рівня прийняття рішень у галузі соціально-економічного розвитку: верхній, середній і нижній.

На верхньому рівні здійснюється макромоделювання процесів соціально-економічного розвитку регіону та формування стратегії соціально-економічного розвитку регіону, вироблення програми розвитку.

На середньому рівні здійснюється вироблення конкретних управлінських рішень щодо економічного регулювання ринку. Інформаційно-аналітична підтримка діяльності муніципальних органів реалізується на рівні мерії, адміністрації, комітетів.

Нижній рівень, з точки зору моделювання – інформаційний, у який збирається інформація з районів, органів управління, комітетів, з адміністративно-господарської служби. На цьому рівні можливе розв’язання

задач аналізу та прогнозування ресурсного потенціалу регіону за допомогою традиційних методів прогнозування. Основні методи опрацювання інформації на нижньому рівні – факторний аналіз, аналіз і прогноз отриманих показників соціально-економічного розвитку на основі згладжування часових рядів.

Аналіз та моделювання соціально-економічних, регіональних систем необхідно виконувати з урахуванням таких характерних особливостей [16]:

- Регіон розглядається як складна слабоструктурована система, методологією дослідження якої є системний аналіз з такими особливостями: наявність великої кількості складних взаємопов'язаних причинно-наслідкових зв'язків між факторами, розглянутими в описі складної системи, результат дії яких не завжди очевидний під час прийняття рішень, необхідність дослідження стохастичних систем в умовах невизначеності, неоднозначності;
- Регіон – динамічна система. Необхідно вивчати динаміку розвитку системи, проводити аналіз процесів росту, з урахуванням загального життєвого циклу регіону і його частин (населення, підприємства, житловий фонд та ін), адаптивної еволюції.

Методологічною основою моделювання процесів соціально-економічного розвитку регіону є системний аналіз, центральною процедурою якого є побудова узагальненої (єдиної) моделі регіону, що відображає всі фактори і взаємозв'язки реальної системи. На практиці це пов'язано з створенням комплексу моделей з розвиненими динамічними та інформаційними зв'язками між моделями усіх рівнів.

Регіон, як об'єкт моделювання характеризується [16, 86]:

- слабкістю теоретичних знань, відсутністю теорії розвитку регіону;
- якісним характером знань про систему, великою часткою експертних знань при описі, структуризації об'єкта моделювання;
- завдання управління регіоном є слабоструктуровані;
- високим рівнем невизначеності вихідної інформації.

Розрізняють внутрішню і зовнішню невизначеність. Внутрішня невизначеність – це сукупність тих факторів, які не контролюються особою, що приймає рішення повністю, але він може чинити на них вплив (наприклад, внутрішня соціально-економічна ситуація, фактори ризику та ін.) Зовнішня невизначеність визначається характером взаємодії з зовнішнім середовищем –- це ті чинники, які знаходяться під слабким контролем особи, яка приймає рішення (екологічна, демографічна, зовнішньополітична ситуація, постачання ресурсів в регіон ззовні і т.п.);

Наслідком цього є те, що результати рішення часто носять якісний характер і дають змогу визначати напрямки розвитку динамічних процесів, виконувати аналіз стійкості динамічних процесів.

Досліджувана соціо-економічна система, якою є регіон, має складну внутрішню структуру, у складі якої можуть бути підсистеми, що декомпонуються: «Населення», «Виробництво», «Невиробнича сфера», «Екологія», «Фінанси», «Зовнішня економічна сфера», – характеризуються ієрархічністю управління та активністю окремих підсистем.

Регіон подається як цілеспрямована і багатоцільова система, що має неоднорідні внутрішні і зовнішні цілі, самостійні підцілі окремих підсистем, систему показників виміру цілей, різноманітні стратегії їх досягнення і т.д.

Система показників соціально-економічного розвитку регіону є складною ієрархічною структурою з безліччю власних показників у кожній з підсистем. У цю систему залежно від завдання управління можуть включатися критерії, що відображають соціальний, економічний, містобудівний та інші варіанти розвитку. У загальному випадку система показників включає інтегрований критерій, що відображає рівень життя населення в регіоні (наприклад, національний дохід на душу населення). На верхньому рівні цієї ієрархічної структури виділяють 3 групи агрегованих критеріїв, що включають [16]:

- узагальнюючу оцінку соціальних параметрів регіону;

- показники, що характеризують об’єктивні економічні (виробничі) умови регіону;
- змінні, що відображають соціальні характеристики невиробничої сфери, залежні від розвитку виробництва.

У свою чергу агреговані показники дають загальну оцінку стану соціально-економічної структури і включають демографічні, соціально-професійні, трудові та суспільно-політичні параметри, а також параметри, що відображають умови життя, праці та побуту населення регіону.

Основними факторами, що діють у розглянутій системі, є: власний ресурсний потенціал регіону (трудові, природні, виробничі, фінансові ресурси), залучені в регіон ресурси (як правило, у вигляді інвестицій і централізованих капітальних вкладень), реальні процеси суспільного виробництва.

Отже, особливостями систем, у яких моделюються регіональні процеси є:

- робота з детальними та агрегованими даними;
- отримання даних з різних джерел як за запитом, так і за розкладом;
- робота з даними різних типів та різних форматів;
- робота з даними, структура яких може бути невідома.

1.2. Аналіз проблем опрацювання різнотипної інформації

Проблема швидкого отримання різнотипових даних (сенсорних числових, текстових документів, графіків тощо) з метою формування на їх основі оперативних рішень постала ще у роки 2-ї світової війни і активно розвивалась для застосування в атомних проектах, управлінні ракетами, навігації, управлінні бойовими діями.

Опрацювання та аналіз таких різнотипових даних використовується для моделювання розвитку подій та ситуацій, а також в системах підтримки прийняття рішень. Започаткували вивчення цієї проблеми фон Нейман, розробки компанії IBM, науковці школи Лєбєдєва С.О. (спеціалізована ЕОМ),

Глушкова В.М. (системний аналіз, теорія конфліктних ігор, проблемно-орієнтовані системи моделювання та опрацювання даних) [1, 5], що призвело до розвитку мов блокового програмування, систем підтримки прийняття рішень.

Проте зміна класу досліджень – від оперативного до аналітичного, поява нових типів даних, необхідність швидкого доступу до них, зумовила збільшення інтересу до проблеми інтеграції та опрацювання даних з метою підвищення якості управлінських рішень. Найвищий пік активності досліджень у сфері інтеграції припадає на 90-ті рр. XX ст. та на наш час [54] у зв'язку з бурхливим розвитком методів Business Intelligence та збільшенням можливостей сховищ даних (збільшення обсягів збережених даних, наявність процедур аналітичного опрацювання даних – OLAP).

Особливістю сучасних досліджень є аналіз не лише типів даних (описів), але й семантики. Особливо активний розвиток засобів для оперативного збору різнотипних даних, завантаження їх у сховище даних, аналізу та прогнозування спостерігається в сферах енергетики та адміністративного керування, нафтогазовому секторі [53].

Схема отримання інформації органами керування регіоном передбачає створення статистичних звітів іншими об'єктами галузі за наперед визначеною формою з покроковим агрегуванням інформації від одного об'єкту до іншого (рис. 1.2).

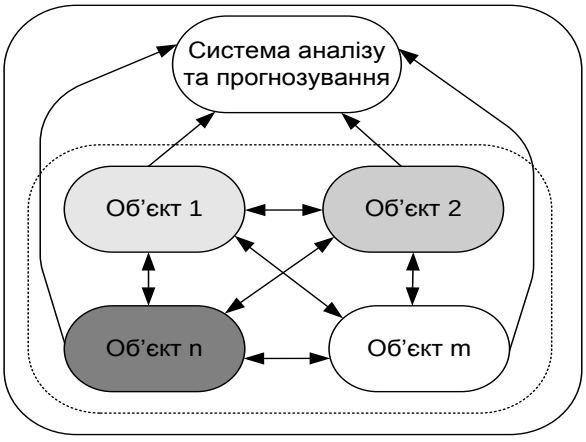


Рис. 1.2. Схема рівноправного обміну даними

Це приводить до того, що особа, яка отримує інформацію, бачить її лише в агреговану вигляді за жорстко визначеними критеріями групування, а деталізована інформація потрапляє зі значним запізненням. Тому рішення, які можуть бути прийняті у такому випадку, недостатньо враховують усі особливості перебігу процесів розвитку регіону.

Процес консолідації даних для аналізу та прогнозування розвитку регіону генерує наступні задачі:

- підвищення оперативності отримання, аналізу та використання інформації, необхідної для підтримання прийняття рішень щодо керування регіоном;
- підвищення якості та дієвості керуючих рішень завдяки оперуванню достовірною інформацією, отриманою безпосередньо з відповідного джерела;
- визначення нових аспектів діяльності регіону завдяки аналізу даних, які не потрапляли у традиційні звіти, і тому не враховувалися при прийнятті рішень;
- своєчасного виявлення негативних тенденцій розвитку з метою їх подальшого усунення.

Інформаційний бум призвів до збільшення кількості даних, накопичених у багатьох предметних галузях у сотні та тисячі разів. До таких областей відноситься і сфера державного управління. Кількість зібраної інформації зростає експоненційно. Так, за дослідженням IDC Digital Universe Study [141], проведеним на замовлення компанії EMC, сумарний обсяг світових даних у 2005 році складав 130 ексабайт, до 2011 року він зріс до 1227 EB, а за останній рік знову подвоївся, досягши 3 ZB (зетабайт). Прогноз, здійснений тим же дослідженням, показує, що до 2018 року обсяг цифрових даних зросте до 7.9 ZB. Розмір окремих баз даних зростає так само швидко і подолав петабайтний

бар’єр. Більшість зібраних даних на даний час не аналізується, або ж проходить лише поверхневий аналіз [52].

Видобування даних є процесом виявлення нових нетривіальних закономірностей з великих масивів інформації. Необхідно зазначити, що прикметник «великий» у застосуванні до даних має тенденцію до постійного зростання значення. Наприклад, за даними Тіма Суонсона кількість операцій, що здійснюється щодня в криптовалюті Bitcoin, перевищила 100 000 операцій (оригінал у «Щоденний обсяг транзакцій в Bitcoin подолав 100-тисячний бар'єр»: [Режим доступу] http://vkurse.ua/ua/business/ezhednevnyy-obem-tranzakciy-v-bitcoin.html). Очевидно, що таку кількість даних неможливо опрацювати без використання автоматизованих засобів. Тому фахівцями розроблено багато методів аналізу даних, пошуку залежностей між ними, прогнозування тощо.

Основними проблемами, які виникають при обробці даних, є відсутність методів аналізу, придатних до застосування через їх різнотипність (для регіону– це і числові дані, і гео-дані, слабоструктуровані звіти тощо), потреба у значних людських ресурсах для підтримки процесу аналізу даних, висока обчислювальна складність наявних алгоритмів аналізу та стрімке зростання обсягу зібраних даних. Це в свою чергу призводить до постійного зростання часу, що витрачається на аналіз даних навіть при регулярному оновленні комп’ютерних засобів, а також – необхідність роботи із розподіленими базами даних, можливості яких більшість існуючих методів аналізу даних не використовують ефективно.

Таким чином, виникає задача розроблення ефективного методу аналізу даних, що може застосовуватись до розподілених баз даних різних предметних областей. Тому для регіону доцільно розробляти методи та засоби консолідації даних та використання їх для аналізу.

1.3. Визначення Великих даних

Концепція Великих даних не нова, вона виникла за часів мейнфреймів і пов'язаних з ними наукових обчислень [136, 138]. Як добре відомо, наукомісткість обчислень завжди було складним завданням. Як правило, вона нерозривно пов'язана з обробкою великих обсягів інформації.

Проте, безпосередньо термін «Великі дані» (Big Data) з'явився порівняно недавно. Він є одним з небагатьох, що має відомий день народження – 3 вересня 2008 р. Тоді було випущено спеціальний випуск найстарішого британського наукового журналу Nature. Журнал присвячений пошукам відповіді на питання: «Як технології можуть вплинути на наукове майбутнє, що відкриває можливості для роботи з Великими даними» [135].

Згідно зі звітом McKinsey інституту під назвою «Великі дані: Наступний рубіж для інновацій, конкуренції і продуктивності», термін «Великі дані» відноситься до наборів даних, розмір яких перевищує ємність звичайної бази даних (БД) для видобування, зберігання, управління і аналізу інформації [135]. Глобальні сховища даних продовжують зростати. Представлений звіт аналітичної компанії IDC під назвою «Digital Universe Study» в середині 2011 року (який був організований компанією EMC) [141] передбачає, що загальний світовий обсяг генерованих і тиражованих даних може досягати в 2011 році близько 1,8 зеттабайт (1,8 трлн гігабайт). Це приблизно в 9 разів більше, ніж те, що було створено в 2006 році [141].

Проте, концепт «Великі дані» означає набагато більше, ніж просто величезні обсяги інформації. Проблема полягає не в тому, що організації генерують величезні обсяги даних, але більшість з них представлені в форматі, який не дуже добре вписується в традиційний структурований формат бази даних. Це веб-журнали, відео, текстові документи, машинний код, або, наприклад, картографічні дані. У результаті, корпорація може мати доступ до величезного обсягу своїх даних і не мати необхідних інструментів для

встановлення зв’язків між цими даними, автоматизованого формулювання конструктивних висновків на основі аналізу цих даних. Крім того, дані зараз оновлюються частіше, і ми маємо ситуацію коли традиційні методи аналізу інформації не можуть опрацювати величезні обсяги даних, що постійно оновлюються, і це, в кінцевому рахунку, прокладає шлях для технологій Big Data.

EWeek подає визначення, запропоноване дослідницькою компанією Gartner: «Великі дані характеризуються обсягом, різноманітністю і швидкою плинністю структурованих і неструктурованих даних в процесорах і пристроях зберігання даних, а також перетворення даних для задач бізнес-консалтингу для підприємств»[52].

Великі дані (Big Data) в інформаційних технологіях за визначенням К. Лінч, Д. Леней – набір методів та засобів опрацювання структурованих і неструктурованих різнотипних динамічних даних великих обсягів з метою їх аналізу та використання для підтримки прийняття рішень [52]. Є альтернативою традиційним системам управління базами даних і рішенням класу Business Intelligence. До цього класу відносять засоби паралельного опрацювання даних (NoSQL, алгоритми MapReduce, Hadoop) [5, 6, 47, 48].

На думку компанії **DCA (Data-Centric Alliance)** під Big Data розуміють не якийсь конкретний об’єм даних і навіть не дані, а методи їх обробки, які дозволяють розподілено обробляти інформацію [141]. Ці методи можна застосовувати як до великих масивів даних (таких як дані всіх сторінок в мережі Інтерет), так і до малих масивів (інформація про денні поступлення товару в магазин).

Визначальними характеристиками для Великих даних є [47] обсяг (volume, в сенсі величини фізичного обсягу), швидкість (velocity в сенсах як швидкості приросту, так і необхідності високошвидкісної обробки та отримання результатів), різноманіття (variety, в сенсі можливості одночасної обробки різних типів структурованих і слабоструктурованих даних).

Хмарні технології підтримують інфраструктуру віртуалізації та її профілювання для конкретних структур даних або для підтримки конкретних наукових робочих процесів [56].

Розмаїтість (Variety) визначається за допомогою [46]:

1) реляційних даних (таблиці / транзакції);
2) текстових даних (Web),напівструктурованих даних (XML);
3) даних на основі графових моделей (соціальна мережа, Semantic Web, RDF);
4) потокових даних;
5) великих публічних даних (онлайн, погода, фінанси і т.д.).

Є такі види вартості (*V*alue) у Великих даних як статистичні дані, події, метадані тощо.

Швидкість (Velocity) (Speed) Великих даних подана як:

1) дані генеруються швидко і повинні бути опрацьовані швидко,
2) он-лайн аналіз даних,
3) підтримка прийняття рішень з неповними даними.

Достовірність (Veracity) – поняття, зворотне до невизначеності, яка виникає через невідповідність даних, їх неповноту, латентність [49].

Аналіз даних в системах територіального управління зводиться до вирішення трьох конкретних завдань:

- соціально-економічна оцінка стану природного середовища в регіоні в даний час і перспективі, розроблення на її основі системи заходів по повному запобіганню чи максимальному пом’якшенню негативного впливу господарської діяльності на навколишнє середовище;
- визначення й врахування можливих наслідків змін у природному середовищі в результаті господарської діяльності і техногенних процесів, їх вплив на спеціалізацію і комплексний розвиток господарства регіону;

- врахування прогнозів еколого-економічних процесів у контексті загального комплексного прогнозу соціально-економічного розвитку регіону шляхом формування ряду критеріїв і обмежень як по ресурсах, так і за допомогою показників якісного стану навколишнього середовища.

Незважаючи на те, що термін був введений в академічному середовищі, первинною була проблема зростання кількості і різноманітності наукових даних. Станом на 2009 рік термін став широко поширений у діловій пресі, а до 2010 року з'явилася перша низка інформаційно-технологічних продуктів і рішень, що стосуються виключно проблем обробки великих обсягів даних. З 2011 року більшість найбільших постачальників інформаційних технологій для організацій в їх бізнес-стратегії використовують концепцію Великих даних, у тому числі IBM, Oracle, Microsoft, Hewlett-Packard , EMC [51, 141].

Завдання, що виникають під час опрацювання, обробки, інтерпретації, збору та організації Великих даних, з'явилися в численних секторах, включаючи бізнес, промисловість, некомерційні організації. Обсяги даних, такі як операції замовника у роздрібній торгівлі, моніторинг погоди, бізнес-аналіз, можуть швидко випереджувати потужність традиційних методів та інструментів аналізу даних. З'явилися нові методи та інструменти, включаючи бази даних NoSQL, MapReduce, обробка природної мови, машинне навчання, візуалізація, придбання, і серіалізація. Усе це стає необхідним, щоб повною мірою усвідомити, що відбувається, коли зростають Великі дані, як вони застосовуються і де починають відігравати вирішальну роль. Також необхідо знати вимоги до існуючих методів розроблення систем і аналізу даних.

Великі дані є терміном, який використовується для ідентифікації наборів даних, з якими ми не можемо впоратися з використанням існуючих методологій та програмних засобів через їх великий розмір і складність. Багато дослідників намагаються розробити методики і програмні засоби для передачі даних або видобування інформаційних гранул з Великих даних [27, 54].

Особливості Великих даних, а саме:

- робота з неструктурованою та структурованою інформацією,
- орієнтація на швидке опрацювання даних,

призводять до того, що традиційні мови запитів виявляються малоефективними для роботи з даними [41].

Тому метою роботи є формальний опис різних моделей даних, виділення операцій та носіїв, а також способів їх сумісного використання.

1.4. Концепції роботи з Великими даними

1.4.1. Концепція NoSQL

Однією з концепцій опрацювання не тільки реляційних даних є NoSQL [132].

Прихильниками концепції мови NoSQL підкреслюється, що вона не є повним запереченням мови SQL і реляційної моделі, що SQL – це важливий і досить корисний інструмент, але при цьому він не може вважатися універсальним. Однією з проблем, яку вказують для класичних реляційних БД, є проблеми при роботі з даними дуже великого обсягу, проектами з високим навантаженням. Основна мета підходу – розширити можливості БД там, де SQL недостатньо гнучкий, і не витісняти його там, де він справляється зі своїми завданнями.

В основі NoSQL є такі фактори:

- нереляційна модель даних,
- розподіленість,
- відкритий вихідний код,
- гарна горизонтальна масштабованість.

У якості одного з методологічних обґрунтувань підходу NoSQL використовується евристичний принцип, відомий як теорема CAP (Consistence, Availability, Partition tolerance – «узгодженість, доступність, стійкість до

поділу»), стверджуючий, що в розподіленій системі неможливо одночасно забезпечити узгодженість даних, доступність (англ. availability, у змісті наявності відповіді по будь-якому запиту) і стійкість до розщеплення розподіленої системи на ізольовані частини. Таким чином, за необхідності досягнення високої доступності й стійкості до поділу не фокусуються на засобах забезпечення узгодженості даних, що забезпечуються традиційними SQL-орієнтованими СУБД з транзакційними механізмами на принципах ACID [133].

Нестроге доведення теореми CAP засноване на простих міркуваннях. Нехай розподілена система складається з N серверів, кожен з яких обробляє запити деякого числа клієнтських застосунків. Під час опрацювання запиту сервер повинен гарантувати актуальність інформації, що міститься у відповіді на запит, що надсилається, для чого попередньо потрібно синхронізувати вміст його власної бази з іншими серверами. Таким чином, серверу необхідно чекати повної синхронізації або генерувати відповідь з урахуванням не синхронізованих даних. Можливий і третій варіант, коли з яких-небудь причин синхронізація здійснюється тільки з частиною серверів системи. У першому випадку невиконаною є вимога стосовно доступності, у другому – узгодженості, у третьому – стійкості до поділу.

Існує чотири категорії NoSQL баз даних [131].

Перша категорія – це Key-Value (Ключ-Значення) бази даних. Це дуже прості по своїй ідеї бази даних. Фактично це дуже великі хеш-таблиці, де кожному ключу поставлене у відповідність значення. Такі бази можуть дуже швидко оперувати колосальними обсягами інформації, але вони мають серйозні обмеження в мові запитів. У якості прикладів Key-Value баз даних можна привести Dynomite, Voldemort, Tokyo, Redis.

Друга категорія – клони Bigtable. BigTable – це база даних розроблена компанією Google для власних потреб. Ця база являє собою велику таблицю із трьома вимірами: колонки, рядки й часові мітки. Така архітектура дозволяє

добитися дуже високої продуктивності, крім того, вона добре масштабується на багато комп'ютерів. Але це нереляційна база даних, і вона не підтримує багато можливостей реляційних баз даних. Зокрема в Bigtable немає операцій з'єднання (join), немає складних запитів і ін. Компанія Google не поширює Bigtable, тому на ринку з'явилося кілька незалежно розроблених клонів цієї бази даних. Зокрема це такі проекти як: Hadoop, Hypertable і Cassandra [28].

Наступна категорія баз даних – це документо-орієнтовані бази даних. Такі бази нагадують Key-Value бази даних [28], але в цьому випадку в метаданих чи словниках бази даних вказано, що собою представляють значення. Зазвичай, значенням є деякий документ або об'єкт, до структури якого можна робити запити. Прикладами таких баз даних є CouchDB і MongoDB [28].

Четверта категорія – це бази даних, побудовані на графах. Такі бази орієнтовані на підтримку складних взаємозв'язків між об'єктами і ґрунтуються на теорії графів. Структура даних у таких базах являє собою набір вузлів, зв'язаних між собою посиланнями. При цьому й вузли, й посилання можуть мати деяку кількість атрибутів. Як приклад можна привести такі бази даних як: Neo4j, AllegroGraph, Sones graphDB [46].

Також існує ще й п'ята категорія, але її не відносять до NoSQL. Мова йде про об'єктно-орієнтовані бази даних. Такі бази даних призначені насамперед для підтримки об'єктно-орієнтованої парадигми програмування. Їх дуже просто використовувати в мовах програмування, у яких підтримується ця парадигма.

Існує кілька механізмів для доступу до даних в NoSQL БД [130].

1. Restful інтерфейси. Це інтерфейс, схожий на основний протокол інтернету – HTTP. У рамках даного підходу передбачається, що кожний об'єкт, яким ми можемо маніпулювати, має свою унікальну адресу. Звертаючись за цією адресою, ми можемо запитувати, створювати, редагувати або знищувати зазначений об'єкт. При цьому на сервері не зберігається ніякого стану, тобто кожний запит опрацьовується незалежно від інших запитів.

2. Мови запитів, відмінні від SQL.

GQL – SQL-подібна мова для Google Bigtable,

SPARQL – мова запитів Семантичного Веба,

Gremlin – мова обходу графів,

Sones Graph Query Language – мова запитів до Sones Graph.

3. API запити.

Google Bigtable Datastore API,

Neo4j Traversal API.

NoSQL можуть запропонувати високий рівень експлуатаційної готовності, коректності й продуктивності. Головною перевагою NoSQL баз даних є продуктивність. Усі NoSQL бази даних перевершують реляційні бази даних у своїй ніші. Якщо ще недавно існував тільки один вид баз даних для всіх випадків – реляційні бази, то сьогодні ситуація змінилася. Для кожної конкретної ситуації необхідно підбирати свою модель даних. Іноді доводиться мати одночасно кілька баз даних, у кожної з яких використовуються її найсильніші сторони. Наприклад, у web-застосуваннях Mongodb застосовується як основне сховище даних, а за допомогою Redis організовується кешування запитів користувача. У результаті ми отримуємо систему з дуже високою продуктивністю й з досить зручними інтерфейсами для розроблювачів. Ще одна важлива перевага NoSQL баз даних полягає в тому, що багато представників цього сімейства сховищ даних реалізовані як проекти з відкритим кодом.

Отже, існування різних категорій баз даних NoSQL вимагає формального опису моделей даних, що ними опрацьовуються.

Загалом порівняння SQL і NoSQL подано в таблиці 1.1 [130].

Таблиця 1.1

Порівняння SQL і NoSQL

	SQL	NoSQL
Зберігання даних	Зберігаються в реляційній моделі в рядках і стовпцях.	Термін NoSQL охоплює множину баз даних, кожна з яких може мати різні моделі зберігання даних. Основні з них: документ, графік, ключ-значення.
Схеми і гнучкість	Кожний запис відповідає визначеній схемі. Схема може бути змінена, але в цьому випадку вона передбачає зміну всієї бази даних. Виконання змін відбувається оффлайн.	Схеми являються динамічною інформацією, яка може бути додана в будь-який момент часу.
Масштабованість	Вертикальна масштабованість.	Горизонтальна масштабованість.
Відповідність ACID	Більшість реляційних баз даних відповідають ACID.	Все залежить від технології. Багато рішень в області NoSQL не дотримуються відповідності ACID на користь масштабованості та продуктивності.

1.4.2. Аналіз програмного забезпечення для роботи з Великими даними

Концепція «Великі дані» досі не дуже добре окреслена, хоча активно використовується для бізнесу та технологій. Аналіз згаданих вище джерел, науково-популярних журналів, і блогів дають змогу виділити наступні фокуси обговорення [18]:

- джерела Великих даних,
- апаратне забезпечення та інфраструктура,
- програмне забезпечення і зберігання,

- інформаційні технології (методи і засоби обробки даних).
- використання Великих даних, бізнес-аналіз.

В якості джерел Великих даних можна виділити пристрої та людей. Приклади перших джерел: національні та міжнародні проекти, такі як Великий адронний коллайдер (LHC) в ЦЕРН, Лабораторія фізики елементарних частинок в Європі, Великий синоптичний оглядовий телескоп на півночі Чилі; промисловість (SCADA, фінанси і т.д.) [21].

Приклади другого типу джерел: соціальні мережі, охорона здоров'я, роздрібна торгівля, особисті дані про місцезнаходження, управління громадським сектором і т.д.

Для збору і обробки Великих даних доцільно використовувати технології хмарних обчислень. Хмарні обчислення – це нова парадигма для розміщення кластерів даних і надання різних послуг локальною мережею або через Інтернет. Хостинг кластерів даних дає змогу клієнтам зберігати та обчислювати величезну кількість даних у хмарі.

Оскільки розмір окремих баз даних зростає швидко і подолав петабайтний бар'єр (наприклад, бази даних соціальних мереж), то онлайн опрацювання в режимі OLAP таких обсягів практично неможливе (рис. 1.3) [21].

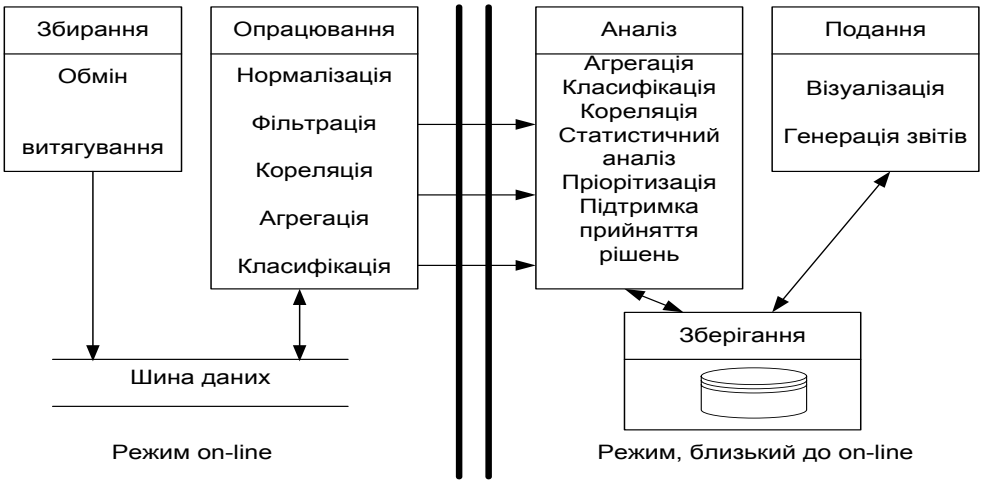


Рис. 1.3. Порівняльна характеристика OLAP та Великих даних

В таблиці 1.2 [129] подано відомості щодо ряду інструментів опрацювання Великих даних з відкритим вихідним кодом, які надаються через інфраструктури хмарних обчислень. Більшість інструментів забезпечується Apache і випущені під ліцензією Apache [129]. Усі ці продукти згруповано за типами задач, що виникають в процесах опрацювання Великих даних.

Таблиця 1.2

Засоби роботи з Великими даними [129]

Засоби для Великих даних	**Опис**
Засоби аналізу даних	
Ambari http://ambari.apache.org	Інструмент веб для надання послуг, управління та моніторингу Apache Hadoop кластерів
Avro http://avro.apache.org	Система серілізації даних
Chukwa http://incubator.apache.org/chukwa	Система колекціонування даних для керування великими розподіленими системами
Hive http://hive.apache.org/	Інфраструктура сховища даних, яка забезпечує агрегацію даних
Pig http://pig.apache.org	Високорівнева мова потоків даних і виконуваний framework для паралельних обчислень
Spark http://spark.incubator.apache.org	Швидкий і генеральний обчислювач для даних Hadoop. Забезпечує просту і виразну модель програмування, яка підтримує широкий спектр додатків, у тому числі ETL, машинного навчання, опрацювання потоків
ZooKeeper http://zookeeper.apache.org/	Високопродуктивна служба координації для розподілених застосунків
Actian http://www.actian.com/about-us/#overview	Забезепечує зберігання сирих даних і готує дані для подальшого аналізу
HPCC http://hpccsystems.com	Забезпечує швидке перетвореня, паралельне опрацювання для застосувань з Великими даними
Засоби Data Mining	
Orange http://orange.biolab.si	Візуалізація та аналіз даних для новачка і експертів

Таблиця 1.2 (продовження)

Mahout http://mahout.apache.org	Бібліотека засобів машинного навчання та видобування даних
KEEL http://keel.es	Еволюційний алгоритм для проблем видобування даних
Засоби соціальних мереж	
Apache Kafka	Платформа з високою пропускною здатністю для опрацювання даних в режимі реального часу
Засоби BI	
Talend http://www.talend.com	Інтеграція даних, управління, інтеграція застосувань, засоби і сервіси для Великих даних
Jedox http://www.jedox.com/en	Функції аналізу, звітності, планування
Pentaho http://www.pentaho.com	Інтеграція даних, бізнес-аналіз, візуалізація даних, прогнозування
Rasdaman http://rasdaman.eecs.jacobs-university.de/	Багатовимірні расторові дані (масив) без обмежень на розмір, наявність мови запитів
Засоби пошуку	
Apache Lucene http://lucene.apache.org	Застосування для повнотекстового індексування і пошуку
Apache Solr http://lucene.apache.org/solr	Повнотекстовий пошук, фасетний пошук, динамічна кластеризація, формати документів типу Word, PDF, просторовий пошук
Elasticsearch http://www.elasticsearch.org	Засіб розподіленого повнотекстового пошуку з веб-інтерфейсом і JSON докуметами
MarkLogic http://developer.marklogic.com	NoSQL і XML база даних
mongoDB http://www.mongodb.org	Крос-платформенна документо-орієнтована система управління базами даних з підтримкою JSON та динамічних схем
Cassandra http://cassandra.apache.org	Маштабована мульти-майстерна база даних без єдиної точки відмови
HBase http://hbase.apache.org	Маштабована розподілена база даних з підтримкою структурованого зберігання даних великого обсягу
InfiniteGraph http://www.objectivity.com	Розподілена графова база даних

1.4.3. Особливості перетворення NoSQL в інші формати

Узагальнюючи результати аналізу відображення систем баз даних NoSQL різних класів в інший формат, можна зробити висновок, що незалежно від класу системи – «ключ-значення», колонкова або документна, існує ряд спільних проблем.

Так як в NoSQL схема може бути присутня неявно, реалізованою в коді програми, то семантично значуще відображення в об’єктну модель автоматично будувати неможливо. Виявлення схеми даних для кожної окремої бази даних вимагатиме залучення експерта. Є сенс будувати відображення на рівні додатку (застосунку), що працює над базою даних, минаючи відображення моделей системи баз даних. Однак ці додатки можуть бути [80]:

- закритими для зовнішніх запитів, які не надають довільного доступу до даних,
- розбитими на частини, які працюють з різними підмножинами даних з одних і тих же баз,
- з однією базою може працювати багато застосунків.

Основна причина проблем, що виникають при семантичному відображенні моделей даних NoSQL в об’єктну модель даних, полягає в їх слабкій структурованості:

- в більшості з них не визначається схема даних в явному вигляді;
- значення в парах «ключ-значення» часто можуть бути довільними і неструктурованими, ключі можуть розглядатися і як елементи структури, так як і дані;
- за відсутності схеми структура пар може змінюватися динамічно;
- фактично, за допомогою операції оновлення можна в будь-який момент ввести нову структуру пар «ключ-значення»;
- бази даних зазвичай не обмежені за кількістю пар в структурах, за довжиною ключів, за вкладеністю пар.

Такі системи можуть використовуватися для вирішення завдань, розрахованих на використання нефіксованих і необмежених слабоструктурованих даних (наприклад, ієрархічних або потокових). З іншого боку, багато програм використовують системи баз даних NoSQL для зберігання досить жорстко типізованих структур. І той, і інший випадки доводиться враховувати при відображенні в об’єктну модель.

Під час аналізу семантики даних звертається увага на використання в ключах фіксованих (імена елементів структур) або нефіксованих (дані) значень. У разі виявлення статичної схеми даних фіксовані значення ключа трансформуються в атрибути типів, нефіксовані – в значення атрибутів.

Під час відображення створюються такі структури [82, 83]:

- фреймові структури для вкладених пар;
- абстрактні типи даних в разі, якщо дані підлягають виявленню чіткої структурі.

Можливі різновиди запитів, які використовуються при вирішенні завдань над даними в моделях NoSQL повинні бути відомі заздалегідь. Дані, за якими необхідно організувати пошук, повинні виявитися ключами в деяких парах «ключ-значення», а дані, які шукають – значеннями, для чого формуються допоміжні пари над основними даними з відповідною структурою даних. Такі пари при відображенні не утворюють нові структури, а відображаються в типи і класи, утворені основними парами. Відповідно до допоміжних пар в цільовій моделі утворюються вторинні ключі. Запити з порівнянням по атрибутам, що є ключами в допоміжних парах, при зворотному відображенні відображаються в операції над цими парами [84, 85].

1.5. Математичні методи подання Великих даних

1.5.1. Багатовимірна модель даних

Одним із способів представлення Великих даних є гіперкуб даних [100, 126].

Основними поняттями багатовимірної моделі даних є:

- гіперкуб даних rel,
- вимір V,
- атрибут A,
- комірка X,
- значення rel(V, A).

Гіперкуб даних містить один або більше вимірів і є впорядкованим набором комірок. Кожна комірка визначається одним і лише одним набором значень вимірів – атрибутів. Комірка може містити дані – значення або бути порожньою.

Під виміром у [140] розуміють множину атрибутів, що утворюють одну з граней гіперкуба. Прикладом часового виміру є список днів, місяців, кварталів. Прикладом географічного виміру може бути перелік територіальних об'єктів: населених пунктів, районів, регіонів, країн та ін.

Отже, V – множина вимірів гіперкуба, $A_{V_i} = \{A_{1_i}, A_{2_i}, ..., A_{k_i}\}, i = 1,...,n$ – множина атрибутів виміру V_i, $A = A_{V_1} \cup A_{V_2} \cup ... \cup A_{V_n}$ – множина атрибутів гіперкуба, $V' \subseteq V$ – множина фіксованих вимірів, $A' \subseteq A$ – множина фіксованих атрибутів. Гіперкуб даних позначимо як множину комірок, що відповідає множинам *V, A: rel(V, A).* Підмножина гіперкуба даних, що відповідає множині фіксованих значень, позначається як $rel'(V', A')$.

Кожній комірці гіперкуба даних $rel \in rel$ відповідає єдино можлива множина атрибутів вимірів $A_{rel} \subset A$. Комірка може бути порожня (не містити

даних) або містити значення показника [140]. Множину комірок гіперкуба *rel(V,A)* позначимо як *V(rel).* Для отримання доступу до даних користувачу необхідно вказати множину необхідних вимірів $V' \subseteq V$ і значень атрибутів $A' \subseteq A$ (фіксувати атрибути). Множину комірок, що відповідають відповідним атрибутам та вимірам, позначимо як $rel'(V',A') | rel' \subseteq rel$.

Ключем виміру є атрибут, який однозначно визначає кортеж (рядок) виміру гіперкуба.

Гіперкуби підтримують ієрархію вимірів і формул без дублювання їх визначень. Набір відповідних гіперкубів складає багатовимірну базу даних (або сховище даних).

Комбінації значень вимірів визначають комірки гіперкуба. Залежно від конкретного застосування (прикладної програми), комірки в гіперкубі можуть розташовуватися як розріджено, так і щільно. Гіперкуби, як правило, стають розрідженими у міру збільшення числа розмірностей і ступеня деталізації значень вимірів.

Багатовимірне представлення даних добре використовувати для задач візуалізації даних та їх аналізу, але у зв'язку з розрідженістю гіперкуба [140] обсяг даних у такому випадку є більший порівняно з реляційним представленням, що є неприпустимим до Великих даних.

1.5.2. Об'єктне подання даних

Інформація про модельовані об'єкти подається множиною складних значень, що зберігаються в ідентифікованих змінних (у подальшому – об'єкти даних). Об'єкт даних ***o*** опишемо як [30, 31]

$$\{id, meta, o\},$$

де *id* – унікальний ідентифікатор зазначеної змінної, *meta* – інформація, що описує структуру зазначеної змінної (метадані), де, зокрема, перелічуються поіменовані атрибути об'єкта даних, ***o*** – складне значення, що описує стан об'єкта.

Об'єкт ***o*** можна описати як $\{id, meta, \{r_1, r_2, ..., r_n\}\}$, де r_i – деяке значення відношень R_i, що виникає у результаті нормалізації значення *o* [80, 98].

Будемо розглядати кожне зі значень r_i як значення іменованого атрибута об'єкта ***o***. Тоді атрибут має розглядатися як змінна відношень, а схема об'єкта даних повинна перелічувати імена атрибутів. Відповідно, об'єкт ***o*** буде описуватися як

$$o = \{id, \{{}_o a_1, {}_o a_2, ..., {}_o a_n\}, \{r_1, r_2, ..., r_n\}\}, \quad (1.1)$$

де r_i – значення відношень R_i із схемою $({}_r a_1{:}D_1, {}_r a_2{:}D_2 .....)$, $r_i = relval(R_i)$, D_j, - не обов'язково різні домени з множини ***D*** доменів. Відзначимо, що в системі ***O*** може одночасно існувати множина не обов'язково різних значень r того самого відношення ***R***, що є значеннями атрибутів різних об'єктів або значеннями різних атрибутів того самого об'єкта.

Розглянемо множину ***O*** усіх вхідних у систему об'єктів. Кожному об'єктові $\boldsymbol{o}_i (\boldsymbol{o}_i \in \boldsymbol{O})$ ставиться у відповідність схема об'єкта $S_i = oSchema(o_i)$, що є власною підмножиною множини ${}_o a, S \subseteq_o a$. Та сама схема S може бути поставлена у відповідність багатьом об'єктам. Класом C, $C \subseteq O$ називають множину об'єктів з однаковою схемою: об'єкти $\boldsymbol{o}_i$ $\{id, meta_i, oV_i\}$ і $\boldsymbol{o}_j$ $\{id_j, meta_j, oV_j\}$ належать до того самого класу C тільки тоді, коли $oSchema(\boldsymbol{o}_i) = oSchema(\boldsymbol{o}_j)$. Отже, можна стверджувати, що між множиною схем і множиною класів існує функціональна відповідність $\boldsymbol{S} = Schema(\boldsymbol{C})$.

Також ми можемо говорити про існування множини ***R*** усіх визначених на множині доменів ***D*** відношень *R*, значення яких є значеннями атрибутів ${}_o a_i$ об'єктів із множини ***O.*** Кожному імені ${}_o a$ ставитися у відповідність одне і тільки одне відношення *R* з множини ***R***, тим самим обмежуючи множину можливих значень атрибута з іменем ${}_o a_i$ значеннями цього відношення *R*. Оскільки мова йде про пару «множина – визначене на цій множині значення», відношення *R* можна позначити як домен атрибута ${}_o a$. Однак, оскільки значення, визначене на цьому домені, являє собою значення відношень (relval) і

також є множиною, будемо називати відношення R *реляційним доменом* атрибута ${}_oa$, $R=rdom({}_oa)$. Можливий випадок, що домени різних атрибутів збігаються $rdom({}_oa_i) = rdom({}_oa_j)$, $a_i \neq a_j$.

Отже, ми можемо говорити про функціональну відповідність між множиною атрибутів $\mathbf{{}_oa}$ і множиною відношень $\boldsymbol{R}$,

$$\boldsymbol{R} = rdom(\mathbf{{}_oa}).$$

Ця відповідність повинна вказуватися в схемі об'кта, що буде перелічувати пари ${}_oa : R$, де $R = rdom({}_oa)$

$$\boldsymbol{o} = \{id,\{{}_oa_1{:}R_1, {}_oa_2{:}R_2,\ldots,{}_oa_n{:}R_n\},\{r_1,r_2\ldots\ldots r_n\}\}.$$

Власне значення об'єкта $\boldsymbol{o}$, що належить до класу C, можна подати як таке відображення $\boldsymbol{o}$ з множини $\mathbf{{}_oa}$ у множину $\boldsymbol{R}$ $\left({}_oa \xrightarrow{Schema(C)} R\right)$, що кожному ${}_oa_i$ з $Schema(C)$ $\left({}_oa_i \in Schema(C)\right)$ буде відповідати значення відношень R_n, що є доменом $oV({}_oa_i)=rval(R_n)$, $R_n=rdom({}_oa_i)$, а кожному ${}_oa_j$, яке не належить S $\left({}_oa_j \notin S\right)$, буде відповідати порожня множина $oV({}_oa_j) = \varnothing$.

Кожне таке відображення відповідає унікальному значенню, що є ідентифікатором цього об'єкта. Тоді множина ідентифікованих об'єктів $\boldsymbol{O}$ - функціональна відповідність між множиною ***id,*** що включає унікальні ідентифікатори id всіх об'єктів $\boldsymbol{o}$ вхідних у множину $\boldsymbol{O}$, і множиною відображень $\left({}_oa\right) \xrightarrow{Schems(C)} R$, що формують власні значення цих об'єктів.

$o = \left(id \to_o V\right)$, де ${}_oV = \left\{{}_oV \mid V = {}_o a \xrightarrow{Schema(C)} R\right\}$.

Припустимо, що значення унікальних ідентифікаторів об'єктів, що входять у множину ***id,*** визначені на домені D_{OID} ($\boldsymbol{id}$ H D_{id}), а імена їхніх атрибутів, що входять у множину $\mathbf{{}_oa}$ – на домені D_A ($\mathbf{{}_oa}$ H D_A), і розглянемо множину доменів D', таку, що $D' - D = \{D_{id}, D_A\}$. Можна показати, що

$O \subset R$, $R' = \left\{R_i' \mid R_i' \subset D_{id} \times D_A \times R_i\right\}, R_i \in R$,.

Оскільки відношення R_i , яке входить у множину $\boldsymbol{R}$, визначене на множині доменів $\boldsymbol{D}$ $(R_i\ ({}_r a_1{:}D_1,\ {}_r a_2{:}D_2\ldots\ldots),\ D \in \boldsymbol{D})$, то R_i' є відношенням, визначеним на множині доменів D' .

Множина $\boldsymbol{O}$ об'єктів $\boldsymbol{o}$ може бути однозначно перетворена до множини значень відношень R', що входять у множину R' , причому між множиною $\boldsymbol{R}$ відношень, що є доменами атрибутів об'єктів $\boldsymbol{o}$, і множиною R' існує така взаємно-однозначна відповідність $R \leftrightarrow R'$ **,** що для будь-якого відношення R_i $({}_r a_1{:}D_1,\ {}_r a_2{:}D_2\ldots\ldots)$, що належить $\boldsymbol{R}$, у множині R' буде існувати відповідне відношення $R'_i({}_r a_{\text{id}}{:}\ D_{\text{id}},\ {}_r a_{oa}{:}\ D_A,\ {}_r a_1{:}\ D_1,\ {}_r a_2{:}\ D_2\ldots\ldots),\ D \in D'$).

Звідси випливає, що

1) об'єкт має унікальний ідентифікатор, і, отже, у системі можуть існувати два об'єкти з абсолютно однаковим власним значенням,
2) значення атрибутів об'єктів є значеннями відношень, що є *множиною* кортежів – іншими словами атрибут можна розглядати як групу повторення.

На рівні збереження *ця ж інформація* подана у вигляді *множини значень відношень* $\boldsymbol{R'}$**,** визначених на доменах з множини D'. Як ми вже сказали, що $D' - D = \{D_{id}, D_A\}$, що, по суті, означає, що інформація про унікальні ідентифікатори об'єктів $\boldsymbol{o_i}$ і про значення, які визначають семантику атрибутів, на рівні збереження існує точно в такому ж вигляді, як і інформація про власні значення цих об'єктів, а саме у вигляді явно заданих значень атрибутів кортежів відношень R'. Отже, рівень збереження цілком описується в термінах реляційної моделі даних. З практичної точки зору цей факт цікавий тим, що він дозволяє реалізувати рівень збереження, використовуючи наявні реляційні СКБД.

Тому об'єктне подання даних за певної модифікації може бути використане для подання інформаційних ресурсів Великих даних. Проте

залишається нерозв’язаною задача трансформації з одних типів представлення даних в об’єктну модель даних.

Таблиця 1.3

Порівняння моделей представлення Великих даних

Назва моделі	Автори	Переваги	Недоліки
Багатовимірна модель	Maté, Alejandro [140]	Добре використовувати для задач візуалізації даних та їх аналізу.	У зв’язку з розрідженістю гіперкуба з неоднорідними даними, їх обсяг збільшується, що є неприпустиме до Великих даних
Об’єктна модель	Chang, Fay [30], Papakonstantinou, Y [31]	За певної модифікації може бути використана для Великих даних.	Нерозв’язаною є задача трансформації з одних типів представлення даних в об’єктну модель даних
Графова модель	Zhou Feng [32]	Добра для аналізу зв’язків між об’єктами.	Велика обчислювальна складність алгоритмів пошуку за умови великої кількості об’єктів

1.6. Аналіз методів прогнозування розвитку регіону

Прогнозування регіональних процесів – найважливіша і невід'ємна частина процесів територіального управління. Від нього в значній мірі залежать економічні, соціальні та екологічні наслідки регіонального розвитку, повнота використання трудових, природних та матеріально-речових ресурсів.

Варіант прогнозу – один з прогнозів, що становлять групу можливих прогнозів об'єкту прогнозування.

Метод прогнозування – спосіб дослідження об'єкту прогнозування, направлений на розробку прогнозу.

Методика прогнозування – сукупність методів і правил розробки прогнозів конкретних об'єктів.

Система прогнозування – система методів прогнозування і засобів їх реалізації, функціонуюча відповідно до основних принципів прогнозування.

Споживач прогнозу – організація, підприємство, установа або окрема особа, що використовує результати прогнозів, а також у ряді випадків формулює завдання на прогноз.

Суб'єктами прогнозування соціально-економічного розвитку є органи державної влади й місцевого самоврядування, корпорації й підприємства, також науково-дослідні й консалтингові організації, окремі експерти, яких залучають для розроблення й упровадження прогнозів.

Об'єктом соціально-економічного прогнозування є соціально-економічні процеси (СЕП) – тобто сукупність економічних і соціальних процесів формування та функціонування соціально-економічної системи, які характеризують динаміку зміни її параметрів на певному рівні господарювання. Практична діяльність суб'єкта стосовно об'єкта прогнозування полягає у тому, щоб певними методами і з використанням певного інструментарію вивчити інформацію про об'єкт або систему і перетворити її в інформацію про майбутнє об'єкта або системи.

Економічні (природні) процеси – це процеси між людиною й природою, які здійснюються за допомогою засобів праці з метою створення матеріальних продуктів виробничих процесів, або інтелектуальних продуктів – інформаційних та інноваційних процесів.

Соціальні (суспільні) процеси – це процеси взаємовідносин між людьми щодо забезпечення виробництва або надбання та споживання створених продуктів. Соціальні (суспільні) процеси формують сферу соціальної економіки, яка охоплює соціальні технології та пов'язані із ними політичні й організаційні процеси.

Взаємодія різноманітних процесів зумовлює утворення комбінованих видів: інформаційно-економічних, виробничо-економічних, інноваційно-економічних (у сфері природних процесів), політико-економічних, соціально-економічних та організаційно-економічних (у сфері суспільних процесів). Зв'язок між природними і суспільними процесами опосередковують процеси ринкового й інституціонального регулювання.

Прогнозування полягає в розробленні й обґрунтуванні можливих варіантів очікуваних змін соціально-економічної ситуації та їхніх наслідків під впливом внутрішніх і зовнішніх чинників. Тим самим прогнозування завершує підготовчий блок функцій управління соціально-економічними процесами, що передбачає розв'язання завдань прогнозування у комплексі з усіма попередніми функціями шляхом розроблення відповідних технологій їхнього виконання.

Економічне прогнозування здійснюють у поєднанні з іншими видами прогнозування: соціальним, політичним, демографічним, науково-технічним, прогнозуванням природних ресурсів тощо. Економічні прогнози – невід'ємна складова прогнозування й планування розвитку суспільних процесів. Зв'язок різноманітних видів прогнозів дістає вияв у послідовності їхнього розроблення. Так, економічні прогнози будують за прогнозами науково-технічного прогресу, природних ресурсів, демографічних процесів тощо.

Кожен прогноз має певні параметри, що не тільки класифікують його, але і надають певну інформацію про нього.

Основними параметрами прогнозів є [21, 88]:

1) достовірність;
2) джерело помилки;
3) обґрунтованість;
4) помилка;
5) період підстави;
6) період попередження;
7) прогнозний горизонт;
8) точність прогнозу.

У процесах прогнозування нерідко використовують авторитетні точки зору експертів досліджуваних галузей. Методи експертних оцінок використовують для аналізу об’єктів і проблем, які цілком або частково не підлягають математичній формалізації, тобто для яких важко розробити адекватну модель. Це пояснюється:

1) невизначеністю та складністю прогнозованих явищ;
2) необхідністю кількісно оцінити події, для характеристики яких бракує необхідної інформації й чіткого знання тенденцій розвитку ситуації;
3) необхідністю враховувати не тільки об’єктивні тенденції розвитку ситуації, а й реакцію учасників подій на рішення, що приймається.

1.7. Постановка задачі дослідження

Основними проблемами, які виникають при обробці даних, є відсутність методів аналізу, придатних до застосування через їх різнотиповість (для регіону – це і числові дані, і геодані, слабоструктуровані звіти тощо), потреба у значних людських ресурсах для підтримки процесу аналізу даних, висока обчислювальна складність наявних алгоритмів аналізу та стрімке зростання

обсягу зібраних даних. Вони призводять до постійного зростання часу аналізу даних навіть при регулярному оновленні апаратних засобів серверів, а також – необхідність роботи із розподіленими базами даних, можливості яких більшість існуючих методів аналізу даних не використовують ефективно.

Таким чином, виникає задача розроблення ефективного методу аналізу даних, що може застосовуватись до розподілених баз даних різних предметних областей. Тому для регіону доцільно розробляти методи та засоби роботи з Великими даними та використання їх для аналізу.

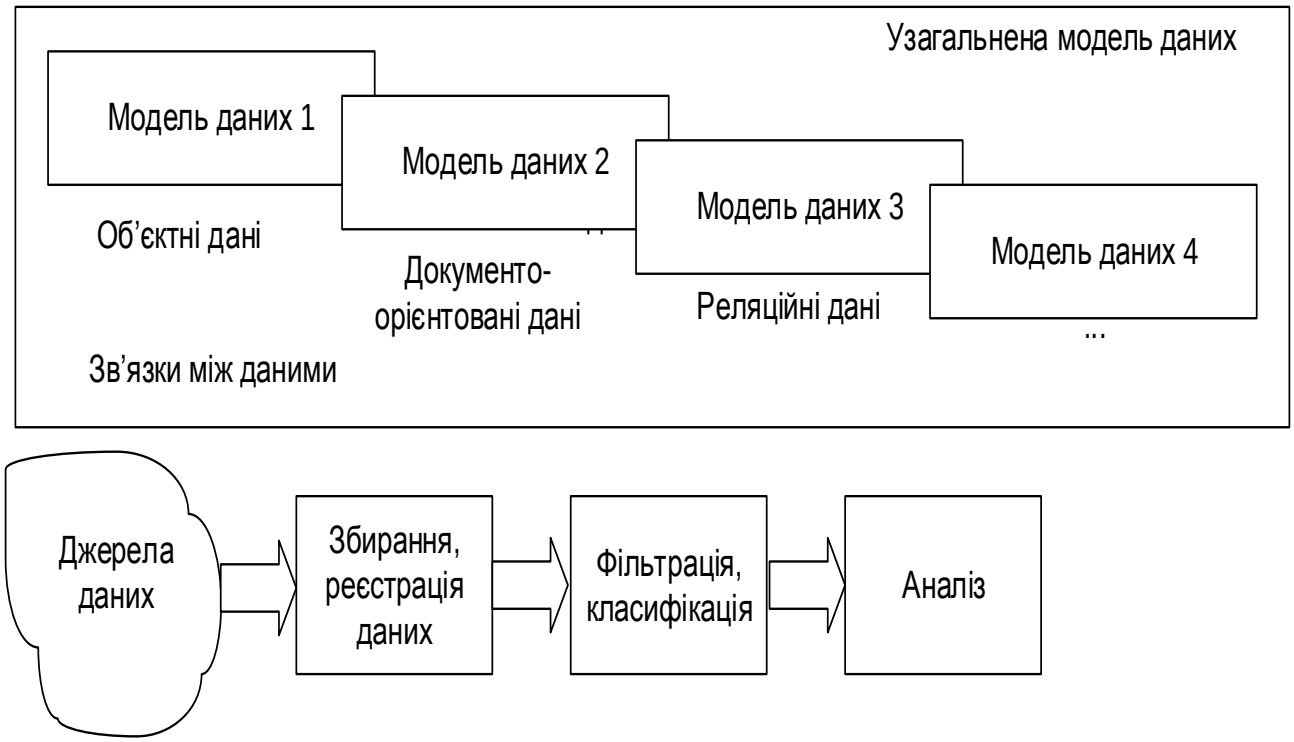


Рис. 1.4. Постановка задачі дослідження

Висновки до розділу 1

1. Визначено складові розвитку регіону і показано, що для цього треба опрацьовувати різнотипні дані.
2. Подано поняття Великих даних та описано їх характеристики. Проаналізовано інформаційні моделі представлення даних великих обсягів та показано їх обмеження. Це призводить до необхідності розроблення моделі представлення Великих даних.

3. Проаналізовано особливості NoSQL-баз даних та принципи перетворення даних в інші формати. Визначено нерозв’язані задачі.
4. Проаналізовано методи прогнозування процесів територіального управління та окреслено їх обмеження.
5. Здійснено постановку задачі дослідження на основі виділення раніше нерозв’язаних задач.

Матеріали розділу опубліковано у [4, 5, 14].

РОЗДІЛ 2.
РОЗРОБЛЕННЯ МОДЕЛІ ВЕЛИКИХ ДАНИХ

У даному розділі здійснено формальну постановку задачі. Подано визначення різних типів даних. Визначено формальний опис Великих даних. Подано моделі асоціацій між сутностями та характеристиками для різних категорій NoSQL баз даних. Використано простір даних для предствлення Великих даних. Подано модель федеративного сховища Великих даних.

2.1. Вибір типів моделей даних для представлення Великих даних

2.1.1. Поняття структурованих та слабоструктурованих даних

Межу між *структурованими* й *неструктурованими* даними можна провести навіть інтуїтивно. Однак, перш ніж перейти до розгляду прикладів такої інформації і її формальних описів, пояснимо відмінність структурованої й слабкоструктурованої інформації докладніше.

Модель даних – це сукупність засобів опису структур даних для додатка або класу додатків. Модель даних містить у собі типи й структури даних, систему операцій, засоби опису обмежень [105]. *Модель структурованих даних* має такі особливості: по-перше, на дані накладаються заздалегідь відомі обмеження за типом й довжиною кожного атрибута, що робить ускладненим, а найчастіше навіть неможливим, модифікацію моделі під вимоги, що змінилися із часом; по-друге, структура даних відома й визначена за допомогою схеми даних, її автоматична зміна в процесі роботи моделі ускладнена. Інтерпретувати дані без знання схеми не представляється можливим. При цьому в процесі розробки схеми необхідно провести формалізацію оброблюваних даних, що унеможливлює автоматизацію коректування схеми в процесі використання моделі. Однак, внаслідок наявних обмежень, модель

структурованих даних має великий набір можливих операцій. Прикладом реалізації моделі структурованих даних може виступати будь-яка реляційна система керування базою даних (СУБД).

Розробка *моделі для неструктурованих даних* є вкрай складним завданням з наступних причин. По-перше, дані, як правило, представлені природньою мовою, що ускладнює роботу з ними. По-друге, повна відсутність визначеної структури накладає серйозні обмеження на можливі операції з даними. Автоматичне виділення структури в таких даних, як правило, не може бути виконане однозначним чином.

Слабоструктурованими даними є будь-які проміжні дані між структурованими й неструктурованими. Такі дані мають певні особливості. По-перше, структура даних може бути неповною, недовизначеною, а також допускати виключення. По-друге, значення скалярних даних представлені у вигляді текстової інформації. По-третє, виникає проблема визначення приналежності даних, тому що не завжди можна однозначно стверджувати про коректність опрацьованого документа.

Модель слабоструктурованих даних повинна враховувати визначені особливості. Виділено основні проблеми, що виникають під час розроблення моделі слабоструктурованих даних. По-перше, при роботі з даними заздалегідь невідомий ступінь їх коректності, і, як наслідок, у моделі необхідний інструментарій для оцінки «правильності» даних. Враховуючи, що в слабоструктурованих даних усі атрибути представлені у вигляді текстової інформації, необхідний досить гнучкий механізм перевірки приналежності даних до конкретного атрибута. По-друге, схема даних може або зовсім не існувати, або не повною мірою відповідати оброблюваним даним. Оскільки працювати з документом, не маючи ніяких уявлень про його структуру, неможливо, виникає завдання виділення схеми з оброблюваних даних, а також її коректування в процесі експлуатації моделі й одержання нової інформації. По-третє, деякі атрибути даних можуть бути або взагалі відсутні, або не повною

мірою задовольняти умовам коректності, заданим для цих атрибутів. Таким чином, у цій моделі повинен існувати інструмент обробки виключень, що дозволяє формувати спосіб запиту до цих даних, ґрунтуючись на заздалегідь заданих критеріях.

2.1.2. Існуючі методи організації зберігання й доступу до слабоструктурованої інформації

Проаналізуємо три основні представлення слабоструктурованої інформації [89]:

1) *OEM (Object Exchange Model*, модель обміну об'єктами);
2) *XML (Extensible Markup Language*, розширювана мова розмітки);
3) *RDF (Resource Description Framework*, фреймворк опису ресурсів).

Ці представлення, як і в науковій, так і практичній літературі даються в описовому, а не формалізованому виді.

Як уже зазначалось в попередньому параграфі, під слабоструктурованими даними розуміють такі дані, які протягом невеликого проміжку часу можуть виявитися недостатньо формалізованими, неповними, або мати структуру, яка може значно змінитися. Як правило, такі дані не можуть бути описані за допомогою якої-небудь незмінної схеми, тому іноді їх називають *schema-less* (такі, що не мають схеми), а також *self-describing* (самоописуваними). Характерною рисою слабоструктурованих даних є те, що описова інформація, яка зазвичай виділяється в окрему схему, є присутньою у самих даних. Досить докладно існуючі методи організації зберігання й доступу до слабоструктурованої інформації розглядалися у роботі [15], а нижче будуть запропоновані формальні моделі для найпоширеніших методів подання слабоструктурованих даних.

Слабоструктурована інформація в OEM-поданні

У рамках OEM-подання [33, 111] – моделі, розробленої раніше всіх інших, поняття слабоструктурованих даних означає скоріше не той факт, що

дані не можуть мати певну структуру, а те, що схема даних піддається доволі частій зміні за кількістю і типом атрибутів сутностей і зв'язків між ними. На етапі розробки ОЕМ-схеми даних, у принципі, немає перешкод для реалізації в реляційних таблицях набагато більшого числа атрибутів даних, аніж потрібно практично, тобто на «майбутнє» їх застосування. Найчастіше при проектуванні схеми даних розробники діють саме таким чином, надалі одержуючи громіздку й надлишкову структуру зберігання, що не піддається адекватному формалізованому опису.

Перерахуємо основні ознаки ОЕМ-подання, що приводять до появи слабкої структурованості БД.

1) Дані потрапляють в базу з гетерогенних джерел. Вони можуть містити пропуски, або дублювання, або мати вкладеність різної глибини.
2) Схема даних може мінятися залежно від семантичної інтерпретації збережених даних. Наприклад, схема даних змінюється залежно від підходу до складання колекції неформатованих текстових документів, присвячених деякій тематиці.
3) Схема даних постійно ускладнюється й вимагає деталізації. Наприклад, часто змінюються умови зовнішнього середовища, граничні значення показників, склад базових і нормативних характеристик, що зберігаються в БД.
4) Схема даних повинна надходити разом із запитами даних, тобто бути вбудованою. Наприклад, різні рівні доступу до конфіденційної інформації не повинні виявляти й розкривати цілком усю структуру БД.

Першим відомим дослідницьким проектом, що практично використовує ОЕМ-подання й спрямований на інтеграцію структурованих даних (представлених реляційними БД), слабоструктурованих даних (зокрема, xml-сторінок) і неструктурованих даних (колекцій файлів текстів) є проект *TSJMMIS* [42].

Архітектура компонентів у зазначеному проекті така:

1) транслятори/конвертори даних;

2) медіатори;

3) менеджери обмежень;

4) класифікатори/екстрактори даних.

Перший компонент призначений для конвертування даних, що надходять із різних джерел до формату єдиної інформаційної моделі проекту *TSIMMIS*. Другий компонент, медіатори, являють собою інформаційні роутери інформаційних запитів, що перенаправляють дані окремим трансляторам, а також об'єднують результати вибірок даних. Третій компонент, менеджери обмежень, виконують перевірку несуперечності й відповідності вихідної інформації вхідному запиту. Четвертий компонент, класифікатори/екстрактори даних необхідні для добування ключових характеристик зі слабоструктурованих (неструктурованих) джерел даних і виділення в них деякого «резюме», тобто деякої системної складової, відповідно до якої їх можна було б віднести до того, або до іншого класу. Обмін інформацією в проекті *TSIMMIS* здійснюється за допомогою самоописової моделі даних *OEM*. Цей проект був неповністю автоматичною системою зберігання й обробки даних, у зв'язку із чим користувачі взаємодіяли з нею за допомогою інтерфейсу *MOBIE (Mosaic Based Information Explorer*), могли переглядати *OEM - об'єкти* й виконувати запити до даних на SQL- подібній мові – *OEM-QL*.

Слабоструктурована інформація в XML-представленні

Умовам *schema-less* і *self-describing* насамперед відповідає *XML*- подання слабоструктурованих даних. Мова *XML* є підмножиною мови *SGML (Standard Generalized Markup Language*, стандартна узагальнена мова розмітки). *SGML* являє собою систему визначення структурованих типів документів і мов розмітки екземплярів документів таких типів. Ця мова дозволяє розділити будь-який документ на дві логічно незалежні частини; одна з них визначає структуру

документа, а інша містить сам текст. Визначення структури називається визначенням типу документа (*Document Type Definition - DTD).* Мова *SGML* уможливлює призначення для будь-якого документа окремо певну структуру й дає можливість авторам документів створювати власні, спеціалізовані структури, тому може стати основою надзвичайно потужної системи керування документами. Проте, ця мова відрізняється значною складністю, тому не одержала досить широкого поширення.

Мова *XML* призначена для виконання аналогічних функцій, але є менш складною і разом з тим більше пристосована для використання в мережі. При цьому дуже важливо те, що мова *XML* зберегла основні переваги *SGML*, розширюваність, структурованість і можливість перевірки правильності документів. Оскільки *XML* є підмножиною *SGML*, то будь-яка система, повністю сумісна з *SGML*, має здатність обробляти документи *XML* (але зворотне твердження не є дійсним). Проте мова *XML* не призначена для заміни *SGML*. До того ж вона не може замінити мови *HTML*, яка також заснована на *SGML* і є додатком цієї мови. У дійсності, *XML* призначена для використання як доповнення до мови *HTML* і повинна забезпечити передачу через *WWW* даних різних типів. Завдяки своїм широким можливостям мова *XML* фактично може застосовуватися не тільки для розмітки тексту, але й для розмітки звуку або зображення, тобто мультимедійних даних. У якості прикладів широко застосовуваних мов, створених на основі *XML*, можна вказати *Mathml (Mathematics Markup Language* – мова математичної розмітки), *SMIL* (*Synchronized Multimedia Integration Language* – мова інтеграції синхронізованих джерел мультимедійної інформації) і *CML* (*Chemistry Markup Language* – мова хімічної розмітки).

Мова *XML* уже фактично стала стандартним засобом обміну даними в індустрії програмного забезпечення. Деякі аналітики припускають, що згодом мовою *XML* буде створюватися й зберігатися більшість документів як в Інтернеті, так і за його межами. Не дивлячись на все це, сама по собі мова не є

моделлю даних, оскільки надає тільки можливості зберігання інформації, і не має інструментів маніпуляції даними, інструментів опису обмежень, способів опису схеми.

Для того щоб використовувати *XML* як повноцінну модель були розроблені наступні рішення. По-перше, мова опису структури *XSD (XML Schema Definition*), що дозволяє описувати схеми даних *XML* документа. *XML Schema* описує спосіб визначення в схемі елемента кожного типу й дає змогу вказати типи даних, пов'язані з кожним елементом. Схема сама є документом *XML*, у якому використовуються елементи й атрибути, що виражають семантику схеми. А оскільки схема – документ *XML,* то її можна редагувати й обробляти за допомогою таких же інструментальних засобів, які застосовуються для обробки описаного нею документа *XML*. Один з найпростіших способів створення схеми *XML* полягає у відстеженні структури документа й визначенні кожного елемента в міру виявлення. Елементи, що містять інші елементи, відносяться до типу складних елементів. По-друге, штучна мова *DTD (Document Type Definition*), що надає можливість створення обмежень і вимог до *XML* документу та його атрибутів. І нарешті, мови запитів, такі як *xpath* і *xquery,* що дозволяють оперувати даними. Методи, засновані на *XML* моделі, не вирішують багато проблем, зв'язаних зі слабоструктурованими даними, тому що не мають необхідного інструментарію відновлення структури документа й засобів опрацювання виникаючих невідповідностей оброблюваних даних до очікуваної структури.

Очевидно, що специфікація *XML Schema* надає повніший і строгіший метод визначення моделі інформаційного наповнення документа *XML,* аніж визначення *DTD.* Але вона все-таки не забезпечує підтримку необхідного рівня семантичної функціональної сумісності. Наприклад, якщо два додатки повинні обмінюватися інформацією за допомогою *XML,* то призначення і зміст застосовуваних при цьому документів повинні бути погоджені з тою структурою даних, яка ними формується. Але для цього необхідно сформувати

модель предметної області з описом даних, які потрібні для того чи іншого додатка. Це дозволяє чітко визначити, які дані повинні передаватися в прямому й зворотному напрямках від одного додатка до іншого. Таку модель зазвичай прийнято описувати в термінах об’єктів або відношень. А оскільки схема *XML* описує тільки синтаксичну структуру документа, то сама модель предметної області може бути представлена у вигляді схеми *XML* багатьма різними способами. Тому неможливо визначити безпосередню відповідність між моделлю предметної області й певною схемою [10]. Ця проблема стає ще складнішою, якщо в обміні інформацією повинні брати участь не два, а кілька додатків. У такому випадку недостатньо просто встановити відповідність між декількома схемами *XML*, оскільки завдання тут полягає не у перетворенні однієї синтаксичної структури в іншу, а у встановленні відповідності між об’єктами й відношеннями, що належать до декількох предметних областей. Таким чином, для розв’язання описаної вище задачі необхідно виконати три етапи:

1) відновити первісні моделі предметних областей зі схем *XML*;
2) визначити відповідності між об’єктами моделей предметних областей;
3) визначити механізми перетворення для документів *XML*, наприклад, за допомогою мови *XSLT* (*Extensible Stylesheet Language Transformations*).

Ці етапи на практиці іноді стають дуже складними, тому може виявитися, що мова *XML* добре підходить для обміну даними тільки між додатками, для яких уже відома модель інформаційного наповнення, але погано підходить для тих ситуацій, коли до обміну даними приєднуються все нові й нові додатки.

Доступ до слабоструктурованої інформації в XML- представленні

Можна виділити такі методи й мови доступу: Xpath (XML Path), Xlink (XML Linking); Xpointer (XML Pointer).

Xpath – це декларативна мова запитів для *XML*, у якій передбачені прості синтаксичні конструкції для адресації окремих частин документа [103]. У мові *Xpath* визначено 13 типів осей (обов’язкових частин), включаючи *ancestor*

(предок), *attribute* (атрибут) і *child* (нащадок). Предикат повинен бути укладений у квадратні дужки й знаходитись після базису. Якщо елемент містить більше одного субелемента, для вибірки конкретного субелемента може застосовуватися конструкція *[position{)= 'positionnumber'],* де *positionnumber* – номер позиції, що починається з 1. У мові *Xpath* підтримується повний і скорочений синтаксис.

Мова *Xlink* дозволяє вставляти в документи спеціальні елементи, за допомогою яких можна створювати й описувати посилання між ресурсами [102]. У ній застосовується синтаксис *XML* для створення структур, що дозволяють описувати посилання, аналогічні простим, односпрямованим гіперпосиланням *HTML,* а також складніші посилання. Посилання *Xlink* можуть ставитися до одного із двох типів: простого і розширеного. Просте посилання з'єднує джерело з ресурсом призначення, а розширене посилання з'єднує будь-яку кількість ресурсів. Крім того, ця мова надає можливість зберігати посилання в окремій БД (її називають базою посилань *Unkbase).* Тим самим надається можливість домогтися в деякій мірі незалежності від місцезнаходження ресурсів: навіть у випадку зміни посилань вихідні документи *XML* залишаються незмінними, і відновлення вносяться тільки в базу посилань.

Мова *Xpointer* надає доступ до значень атрибутів або до вмісту елементів, що перебувають у будь якому місці документа [104]. Будь-який покажчик *Xpointer,* за суттю, являє собою вираз *Xpath, що* входить в ідентифікатор *URI.* Крім усього іншого, мова *Xpointer* дозволяє сформувати посилання на розділи тексту, вибрати конкретні елементи або атрибути й перейти з одного елемента на інший. Ця мова дає можливість також здійснити вибірку інформації, що знаходиться більше ніж в одному наборі вузлів, тоді як за допомогою мови *Xpath* це завдання виконати неможливо. Крім визначення вузлів, у мові *Xpointer* визначаються також точки й діапазони, які в комбінації з вузлами дозволяють позначити місцезнаходження даних. Точка – це позиція в документі *XML,* а діапазон позначає всю структуру й інформаційне наповнення *XML* між

початковою й кінцевою точками, причому й та, і інша можуть перебувати в середині вузла.

Слабоструктурована інформація в RDF-представленні

Інфраструктура опису ресурсів (*Resource Description Framework - RDF),* розроблена під егідою *W3C,* являє собою інформаційне середовище, яке забезпечує кодування, обмін і повторне застосування структурованих метаданих [101]. Ця інфраструктура забезпечує функціональну сумісність метаданих різних додатків за рахунок застосування таких проектних механізмів, які дозволяють створювати загальноприйняті угоди по семантиці, синтаксисі й структурі документів. Інфраструктура *RDF* не визначає семантику кожної розглянутої предметної області, а надає можливість створювати в міру необхідності елементи метаданих для таких предметних областей. В інфраструктурі *RDF* у якості загального синтаксису для обміну й обробки метаданих застосовується мова *XML*. За допомогою засобів мови *XML* створюються моделі даних *RDF,* структура яких забезпечує опис семантики, а також дозволяє створювати однаковий опис і забезпечувати обмін стандартизованими метаданими.

Найпростіша модель даних *RDF* складається із трьох об’єктів.

1) *Ресурс.* Ресурсом є все, що може мати ідентифікатор *URL,* наприклад, Web-сторінка, ряд Web-сторінок або навіть частина Web-сторінки, така як елемент *XML*.
2) *Властивість.* Це – конкретний атрибут, який служить для опису ресурсу. Наприклад, атрибут *Author* може використовуватися для опису особи, що підготувала конкретний документ *XML*.
3) *Оператор.* Це – конструкція, що складається з комбінації ресурсу, властивості й значення. Компоненти оператора *RDF* прийнято називати «суб’єктом», «предикатом» і «об’єктом».

Схема *RDF* дозволяє представити інформацію про класи, що входять у схему, у тому числі про їхні властивості (атрибути) і про зв’язки між ресурсами

(класами). Інакше кажучи, механізм визначення схеми *RDF* надає базову систему типів для використання в моделях *RDF*, аналогічно схемі *XML*.

Схеми *RDF* надають також можливість визначити деяку частину обмежень, зокрема вказати необхідну кардинальність і визначити припустимі властивості екземплярів класів. Для визначення схеми *RDF* застосовується декларативна мова, створена під впливом ідей з області представлення знань (наприклад, семантичних мереж і логіки предикатів), а також моделей представлення схем БД, таких як двійкові реляційні моделі й моделі даних на основі графів.

Слабоструктурована інформація в документо-орієнтованому представленні

Грунтовний виклад методів документо-орієнтованого зберігання подано в роботі [27, 120]. В основі цих методів лежать колекції даних, тобто набори, у яких збираються однотипові за деякими ознаками документи. Колекції призначені для узагальнення однотипних документів, підтримки схем даних документа й забезпечення підтримки групових операцій над документами. Для кожної колекції може бути визначена своя схема даних, свій пошуковий індекс і додаткова інформація про мітки. Якщо проводити паралель із реляційними моделями даних, то колекція – це аналог кортежів (таблиць), з єдиною відмінністю, що про зв’язки між колекціями мова навіть не йде. Документ сам по собі представляє одиницю зберігання даних і кожний з них не залежить від інших. При роботі зі слабоструктурованими даними тип подання інформації може мінятися від документа до документа в рамках однієї колекції. Це показує ступінь відмінності з реляційними моделями, де тип видачі інформації із БД визначається заздалегідь, при проектуванні схеми даних.

Для ідентифікації документів кожному документу присвоюється універсальний ідентифікатор. У якості ідентифікатора можна використовувати рядок, що має достатню складність і довжину для забезпечення унікальності. Найкращим варіантом вважається застосування хеш-функцій *MD5*. Над

документами в колекції визначені такі операції:

1) Додавання нового документа. Головна особливість операції додавання полягає в тому, що знання схеми даних для даної операції не потрібно, більше того, колекції в момент вставки документа може не існувати. У момент вставки, модель повинна перевірити документ, що додається, скорегувати або створити схему даних, і повинна зберегти дані про операцію для подальшої статистичної обробки.
2) Відновлення документа. Вона визначена тільки цілком для документа. Таке штучне обмеження дозволить перенести контроль атомарності на рівень цілого документа. Відсутнє поняття «ізоляції». Будь-які дані, які зчитуються одним користувачем, можуть паралельно змінюватися іншим клієнтом. За рахунок відсутності механізму блокувань і контролю цілісності даних виходить збільшення швидкості роботи БД.
3) Операція видалення документа. Операція може бути виконана тільки за будь-якою пошуковою умовою.

Документи мають значення елементи-атрибути. З них формується весь документ, і вони, за суттю, є найменшою одиницею даних, якою можна оперувати за допомогою документо-орієнтованої моделі. На відміну від класичних реляційних моделей атрибути не мають строгої типізації, тому що оброблювані дані не мають явно вираженої структури. Наприклад, той самий атрибут у різних документах однієї колекції може бути представлений як текстом, так і числом. Значення в атрибутах не мають обмежень на довжину й можуть бути змінені тільки спільно з усім документом; таким чином, операція відновлення (зміни) вмісту одного атрибута не визначена. Для контролю коректності даних, що вводяться, визначені спеціальні функції-валідатори, що беруть на себе завдання з контролю обмежень, наявних для атрибутів.

Функції-валідатори призначені для організації гнучкої типізації даних у БД, тому що слабоструктуровані дані найчастіше представлені неформальною

мовою, у зв'язку із чим виникає завдання визначення типів даних. При цьому найчастіше неможливо однозначно визначити тип даних атрибута, тому завдання його розпізнавання перебирають на себе зазначені функції, що аналізують зміст атрибута та повідомляють наскільки (якою мірою) даний атрибут відповідає шуканому типу даних.

Елементи реалізації ідеї слабоструктурованої обробки й зберігання даних є в безсхемних БД, що відносяться до типу *NoSQL(Not Only SQL,* не тільки *SQL)* [28] систем. Їхньою особливістю, зокрема, є горизонтальне масштабування, сховища даних та підтримка пошуку й індексування по довільних полях, а в деяких БД є можливість формування будь-яких запитів вибірки даних. Найпростішим способом реалізації слабоструктурованого зберігання даних є динамічне сховище ключів і значень, яке реалізовано в БД *Redis* і *Riak.* Іншим підходом до забезпечення можливості динамічної зміни структури БД є стовпчикова реалізація зберігання даних (протилежно до рядкової в реляційних БД), при якій є можливість визначення різної кількості стовпців для різних рядків. За таким принципом влаштовані БД *Hbase, Cassandra, Hypertable.*

До таких, зокрема, відносять БД *Mongodb, Couchdb.* У БД *Mongodb* документ буде ставитися й зберігатися в якій-небудь колекції, а форматом зберігання служить структура мовою *JSON.* Схема БД *Mongodb* повністю змінювана, використовує технологію *Google Mapreduce*, що допускає побудову широкого спектра індексів документів: унікальних, складних, геопросторових і вкладених. Для стійкості й надійності зберігання даних застосовується атомарність операцій, журналювання, технологія асинхронної реплікації із сегментацією по декількох наборах реплік. Безсхемна БД *Couchdb* також використовує масово-облікові функції *mapreduce,* інтерфейс *REST API* для безпосередньої обробки вилучених даних через *HTTP* протокол, а також формат опису даних *JSON.*

Поряд із привабливими можливостями документо-орієнтованого зберігання даних є й особливості, які не дуже приймають розробники

автоматизованих систем на основі БД. Серед таких: відсутність транзакцій, неможливість автоматичного приведення типів даних, ускладнена робота з масивами даних у відношенні їх сортування й фільтрації, вимога приведення даних до певного формату й відсутність інших звичних функцій БД.

Великі обсяги даних вимагають нових підходів і методів добування знань і даних з них. У зв'язку з тим, основні наукові положення пов'язані з розробкою графових моделей зберігання слабоструктурованої і нечіткої інформації. Подальші положення стосуються цих методів і їх реалізацій. Тому необхідно виконати докладний аналіз місця графових моделей даних у сучасних аналітичних системах і платформах [16].

За оцінками інформаційно-аналітичної компанії *IDC (International Data Corporation*) обсяги «цифрового всесвіту» (*Digital Universe)* [66] до 2020 року можуть досягти 40 зеттабайт, тобто 40 трильйонів гігабайтів, з яких до 80% буде становити так звана погано-, слабоструктурована інформація, що циркулює у вигляді інтервальних медіа-потоків, інформації соціальних мереж, медіа-мереж, мережних мобільних пристроїв і так далі. Графічна ілюстрація, за даними *IDC,* наведена на рис. 2.1, показує співвідношення структурованої й слабоструктурованої інформації в сучасній *Digital Universe.*

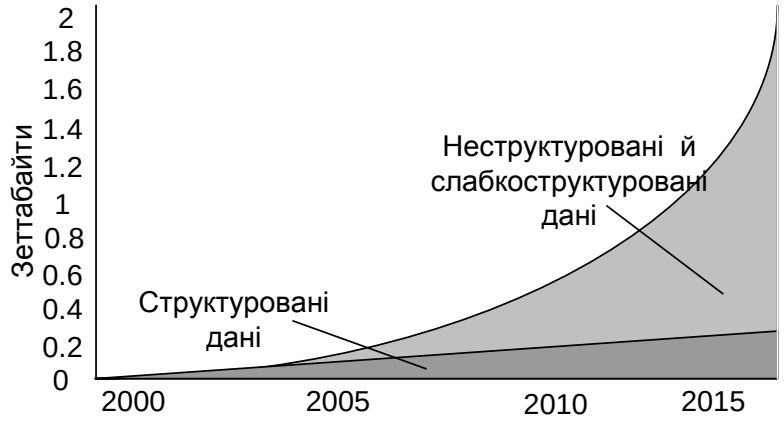


Рис. 2.1. Співвідношення структурованої й слабкоструктурованої інформації в *Digital Universe*

Превалювання слабоструктурованих даних над структурованими, як і

мультиструктурної моделі даних над реляційною моделлю, відзначається й в аналітичних матеріалах досліджень IT-ринку компанією *Gognizanl*, рис. 2.2.

Основними постачальниками, оброблювачами й сховищами даних у сучасній *Digital Universe* є розподілені мережні системи й пристрої різного призначення.

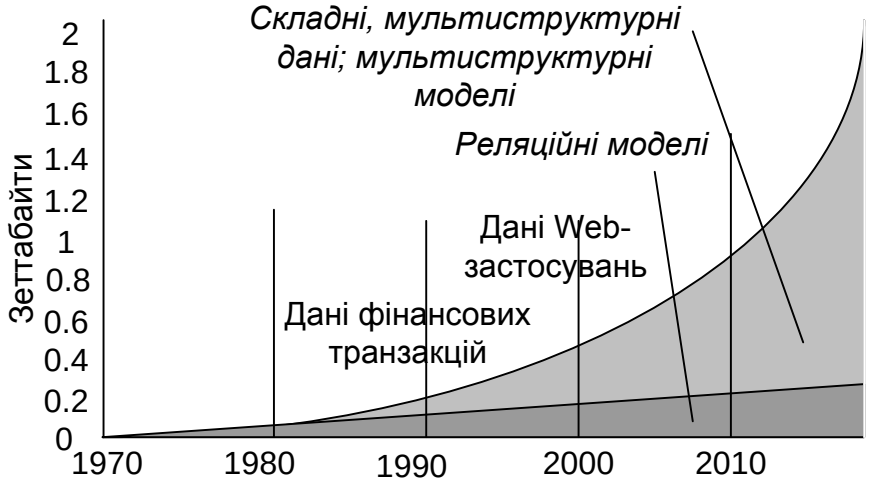


Рис. 2.2. Співвідношення мультиструктурних моделей даних і реляційних моделей

2.2. Формальний опис структури Великих даних

Яскравим прикладом Великих даних є масив даних, що описує функціонування регіону:

1) **обсяги інформації у петабайти:**

   оцифровані книжки бібліотек (у Львівській області налічується 1356 бібліотек з бібліотечним фондом у 18227,4 тис. примірників, що примірно дорівнює 17,4 петабайт інформації, кількість GPS-сигналів від автомобілів кожної транспортної компанії, тощо);

   статитичні дані офіційних установ у Львівській області (1849 сіл, 44 міста, 34 смт. та 1 селище) – щорічно 1 терабайт інформації;

2) **необхідність обробки структурованої** (бази даних відділів охорони здоров’я, податкової інспекції тощо) **та неструктурованої інформації**

(статистична звітність (населення та міграція, ринок праці, освіта, охорона здоров'я, доходи та умови життя, соціальний захист, населенні пункти та житло));

3) **он-лайн аналіз даних** для швидкого доступу до детальних даних.

Отже, у підсумку існує:

- великий набір сутностей: особи, місця, організації (фізичні, юридичні), дати, природні ресурси (річки, ліси, озера), рекреаційний фонд (історичні пам'ятки, санаторії), законодавчі акти та звіти;
- величезна база даних особливостей: документи для інтелектуального аналізу даних, онтологічні терміни, словники даних, які дозволяють зв'язати деякі об'єкти.

Грунтуючись на цій інформації, ми повинні вирішити, які сутності і як пов'язані між собою.

Отже, інформаційна інформаційних ресурсів Великих даних – це $BigD = \langle e, f, a \rangle$, де сутності $e \in E$, характеристики $f \in F$, асоціації $a \to n_{e,f}$ між сутностями e та характеристиками f.

Формально можемо поділити усі об'єкти на такі категорії:

- сутності e,
- характеристики f,
- асоціації між сутностями e та характеристиками f.

Наприклад:

- Ім'я e згадується у документі f,
- Термін f' з'явився у документі e'.

Нехай також визначено:

- множину сутностей E;
- множину характеристик F;
- для кожних e і f зазначено номер асоціацій між e і f як $n_{e,f}$.

Загальна кількість сутностей визначається як $|E|$, загальна кількість характеристик є потужністю множини $F: |F|$. Також опишемо:

- для кожної характеристки f множину $e(f) = \{e \in E : n_{e,f} > 0\}$ усіх асоційованих з f сутностей;
- для кожної сутності e множину $f(e) = \{f \in F : n_{e,f} > 0\}$ усіх асоційованих з e характеристик.

Опишемо ці якісні представлення у кількісному вигляді.

Для цього скористаємося аналогом опису міри TF-IDF у текстових документах [126].

У подібних ситуаціях, коли у нас є кілька сутностей, пов'язаних з харакетристикою, використаємо кількісне представлення інформації, тобто кількість бінарних запитань (так, ні), які необхідно задати, щоб знайти потрібний об'єкт. Загалом, якщо ми знаємо, що невідомий об'єкт належить множині, що складається з N елементів, то ми можемо розділити цей набір на дві половини і, задаючи двійкові питання, з'ясувати, до якої половини належить шуканий об'єкт. Отже, тоді кількість об'єктів становитиме $\frac{N}{2}$. Продовжимо далі таку ж процедуру: задамо друге питання, для чого поділимо виділену половину ще на дві половини. Отже, після двох запитань матимемо $\frac{N}{4}$ об'єктів, серед яких є шуканий. Після трьох запитань матимемо $\frac{N}{8}$. Загалом, після відповіді на q бінарних запитань матимемо множину з $N \cdot 2^{-q}$ елементів, що містить необхідний об'єкт [123, 124].

Коли множина складатиметься з одного елемента, ми точно визначимо необхідну нам альтернативу. Кількість бінарних запитань для пошуку характеристики для N альтернатив: $N \cdot 2^{-q} = 1$, або $q = \log_2(N)$.

Таким самим чином можна описати сутності. Маємо $|E|$ сутностей з кількістю інформації $\log_2(|E|)$. Коли ми знаємо, що якась сутність асоційована з характеристикою (маємо $|e(f)|$ сутностей асоційованих з характеристикою f), то кількість питань рівна $\log_2(|e(f)|)$. Таким чином, той факт, що сутність e пов'язана з характеристикою f, дозволяє зменшити кількість питань до

$$k = \log_2(|E|) - \log_2(|e(f)|) = \log_2\left(\frac{|E|}{|e(f)|}\right). \quad (2.1)$$

Крім того, ефект кількох асоціацій може бути описаний шляхом підрахунку кількості додаткових бінарних питань, які ми можемо задавати, щоб і надалі знати асоціацію з необхідною сутністю. Почнемо з $n_{e,f}$. Кожне бінарне запитання зменшує кількість об'єктів на половину; q питань зменшують кількість до $n_{e,f} \cdot 2^{-q}$. Ми продовжуємо мати асоціацію до того часу, поки кількість об'єктів ≥ 1. Найбільша кількість q, для якої ми ще маємо асоціацію визначиться як $n_{e,f} \cdot 2^{-q} = 1$, звідки $q = \log_2(n_{e,f})$. Додавання додаткового запитання визначається як $1 + \log_2(n_{e,f})$ і означає абсолютну асоціацію.

Загальна важливість характеристики f для сутності e визначається як $\log_2\left(\frac{|E|}{|e(f)|}\right)$ з фактором важливості $1 + \log_2(n_{e,f})$. Результуюча кількість питань визначається як

$$I(e, f) = (1 + \log_2(n_{e,f})) \cdot \log_2\left(\frac{|E|}{|e(f)|}\right). \quad (2.2)$$

Ця формула є одним з варіантів в термінах частоти термінів – так званою зворотною частотою документа tf-idf [6, 127]. Для кожної сутності e маємо кількість питань $I(e, f)$ для різних характеристик f. Значення важливості необхідно нормалізувати:

$$V(e,f)=\frac{\left(1+\log_2(n_{e,f})\right)\cdot\log_2\left(\frac{|E|}{|e(f)|}\right)}{\sqrt{\sum_{j\in f(e)}\left(\left(1+\log_2(n_{e,j})\right)\cdot\log_2\left(\frac{|E|}{|e(j)|}\right)\right)^2}}. \qquad (2.3)$$

Для кожної сутності e є вага $V(e,f)$. Таким чином, в якості міри близькості між двома об'єктами E_1 і E_2, ми можемо вважати відстань між відповідними векторами $(V(e_1,f),V(e_2,f),...)$.

У звичайній Евклідовій відстані $d(a,b)=\sqrt{(a_1-b_1)^2+...}$ додаються квадрати різниць. Отже, для кожної ваги $V(e,f)$, що репрезентує кількість відповідей «так»-«ні», матимемо

$$d(e_1,e_2)=\sum_{f\in F}|V(e_1,f)-V(e_2,f)|. \qquad (2.4)$$

Ця відстань залежить від кількості характеристик: наприклад, якщо на додаток до документів, ми зберігаємо їх копії, відстань збільшується вдвічі. Щоб уникнути цієї залежності, відстань $d(e_1,e_2)$, як правило, нормалізується в інтервалі [0,1] через ділення на максимально можливе значення цієї відстані.

Як ми можемо оцінити найбільше можливе значення цієї відстані? Загалом, коли ми не знаємо істинного значення A і B двох невід'ємних величин, а ми знаємо тільки верхні межі цих величин $\bar{a}$, $\bar{b}$, то найбільша можлива величина різниці $|\bar{a}-\bar{b}|$ дорівнює $\max(\bar{a},\bar{b})$. Тоді [2]:

- якщо $\bar{a}\le\bar{b}$, тоді $|\bar{a}-\bar{b}|=\bar{b}-\bar{a}\le\bar{b}$ і тому, $|\bar{a}-\bar{b}|\le\max(\bar{a},\bar{b})$;
- якщо $\bar{b}\le\bar{a}$, тоді $|\bar{a}-\bar{b}|=\bar{a}-\bar{b}\le\bar{a}$ і тому, $|\bar{a}-\bar{b}|\le\max(\bar{a},\bar{b})$.

В обох випадках маємо $|\bar{a}-\bar{b}|\le\max(\bar{a},\bar{b})$.

Межа $\max(\bar{a},\bar{b})$ досягається у випадках:

якщо $\bar{a}\le\bar{b}$, то для $a=0,b=\bar{b}$;

якщо $\bar{b}\le\bar{a}$, то для $a=\bar{a},b=0$.

Якщо $\bar{a} = V(e_1, f)$ і $\bar{b} = V(e_2, f)$, то максимально можлива відстань між об'єктами $|V(e_1, f) - V(e_2, f)|$ може бути оцінена як $\max(V(e_1, f) - V(e_2, f))$. Порівнюючи сутності по кожній з характеристик, отримаємо відстань між ними, подану як

$$D(e_1, e_2) = \frac{\sum_f |V(e_1, f) - V(e_2, f)|}{\sum_f \max(V(e_1, f), V(e_2, f))}. \quad (2.5)$$

На рис. 2.3 показано використання моделі «сутність-характеристика» для порівняння схем даних реляційної бази даних та NoSQL бази даних, які описують частину предметної області «Курортний комплекс» та містять інформацію про готель та клієнтів. По середині міститься інформація про сутності – Клієнт та Готель, а також асоційовані з цими сутностями характеристики. У першу чергу визначається, у скількох джерелах характеристики асоціюються з вказаними сутностями. Далі ця інформація може бути використана для порівняння схем за сутностями.

Введемо поняття моделі асоціацій між об'єктами та характеристиками для різних категорій NoSQL баз даних.

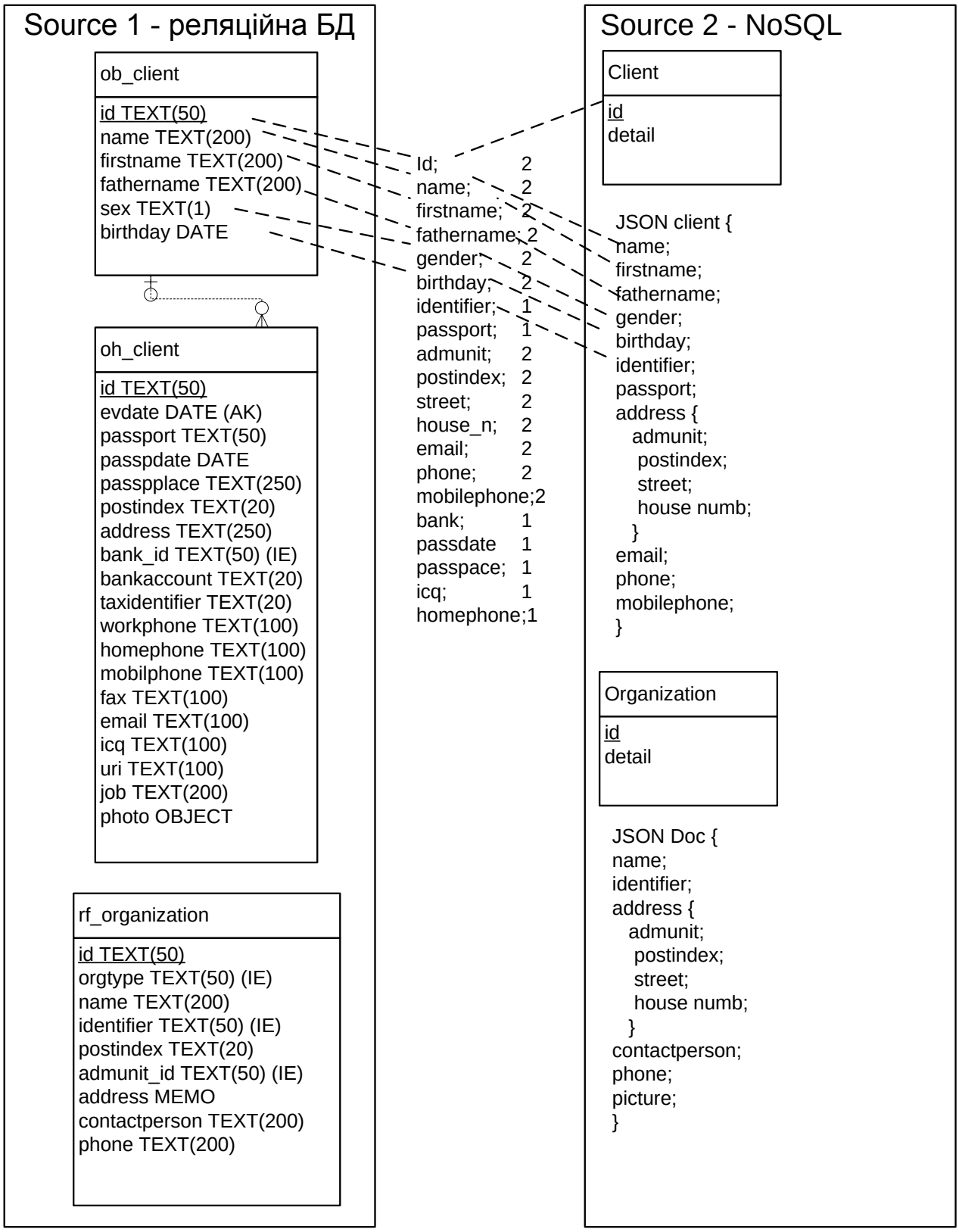


Рис. 2.3. Порівняння схеми реляційної бази даних та NoSQL бази даних з допомогою моделі «сутність-характеристика»

2.3. Моделі асоціацій між сутностями та характеристиками для різних категорій NoSQL баз даних

Носій даних у моделі «ключ-значення» (інша назва – колонкова БД) описується кортежами виду:

$$KV = \{< k, v >\}, \tag{2.6}$$

k – ключ, який приймає унікальні значення у кожній парі, v – значення, що відповідає цьому ключу. Ключі можуть бути складеними (major або minor), значення підтримує практично необмежену семантику, $e \leftrightarrow k$; $f \leftrightarrow v$.

Сигнатура моделі виглядає як :

$$O = \langle \pi, \sigma \rangle, \tag{2.7}$$

де π – операція проекції за атрибутами (ключ або значення), σ – селекції атрибутів (вибір значення за ключем, ключів за значенням, ключів за значенням предків). Перераховані операції відносяться до категорії читання [82 , 83].

Приклади реальних операцій читання:

1) get(key);
2) multiGet:
   - MultiGetIterator, StoreIterator (за major key);
   - Subrange (keyFirst, keyLast);
   - Depth.CHILDREN_ONLY;
3) multiGetKeys:
   - Subrange (keyFirst, keyLast);
   - Depth.CHILDREN_ONLY.

Прикладом СУБД колонкового типу є Cassandra.

Для версійного розподіленого зберігання великих обсягів даних була спроектована модель, що використовується у системі BigTable компанії Google:

- неповна реляційна модель даних,
- підтримка динамічного контролю над розміщенням даних.

Основа моделі даних Bigtable проста: рядки, стовпці і тимчасові мітки.

$$BigTable = \{< r, c, t >\}. \tag{2.8}$$

У базі пошукача іменами рядків можуть слугувати адреси документів з інтернету, а іменами стовпців – особливості цих документів (наприклад, зміст документа може зберігатися в стовпці «content:», а посилання на дочірні сторінки – в шпальтах «anchor:»). Інший приклад – карти Google, що складаються з мільярдів зображень, кожне з яких деталізує ту чи іншу географічну ділянку планети. У Bigtable карти Google структуруються таким чином: кожному рядку відповідає один географічний сегмент, а стовпцями є зображення, з яких цей сегмент складається, у різних стовпчиках зберігаються зображення з різною деталізацією:

$$\begin{matrix} f \leftrightarrow r(t) \\ e \leftrightarrow c \end{matrix}. \tag{2.9}$$

Якщо в кількох стовпцях зберігаються дані одного типу, такі стовпці, згідно моделі Bigtable, утворюють сімейство: $colF = \{c_i, c_j \mid dom(c_i) \in T \wedge dom(c_j) \in T\}$. Використовувати сімейство стовпців зручно хоча б для того, щоб стиснути однорідні дані, тим самим зменшивши обсяг. Саме сімейства стовпців є одиницею доступу до даних.

Рядки Bigtable (їх максимальна довжина може досягати 64 кілобайти) теж важливі. Операція звернення до рядка є атомарною (це означає, що, поки одна програма звертається до рядка, жодна інша не має права змінювати дані в сімействах стовпців цього рядка). А ще рядки зручно сортувати. У прикладі з URL документа, зробивши його запис реверсивним, легко впорядкувати всі рядки за іменем домена третього рівня.

Вміст сторінок в інтернеті постійно змінюється. Щоб врахувати ці зміни, кожній копії даних, що зберігаються в стовпці, присвоюється тимчасова мітка (timestamp). У Bigtable тимчасовою міткою слугує 64-розрядне число, яке може кодувати час і дату так, як це потрібно клієнтським програмам. Наприклад, timestamp для копій веб-сторінки в стовпці «contents:» є датою і часом

створення цих копій. Використовуючи тимчасові мітки, додатки можуть задати в Bigtable пошук, наприклад, тільки найостанніших копій даних.

Отже, для предметної області будь-якого сервісу Google можна створити власну карту даних Bigtable, що містить довільне число рядків і унікальний для цієї предметної області набір сімейств стовпців. Неминучі повтори даних у стовпцях упорядковуються за тимчасовими мітками. Усе це вказує на повну відсутність підтримки властивостей ACID.

Але головною перевагою цього підходу є те, що таку базу неважко порізати на незалежні шматочки і розподілити на множині серверів. Відсортовані за алфавітом рядки діляться на діапазони, які іменуються «Таблет» (tablet) – несамостійними таблицями. Оскільки рядки в кожному Таблеті відсортовані за ключовим іменем, то клієнтським додаткам дуже просто знайти потрібний таблет, а в ньому – потрібний рядок.

В цій моделі ключ ідентифікує рядок, який містить дані, що зберігаються в одному чи декількох сімействах стовпчиків. В рамках таких сімейств кожен рядок може мати багато значень стовпчиків. Значення в кожному стовпці містять мітку часу, тому декілька значень-відповідностей між рядком і стовпчиком можуть знаходитися в рамках одного сімейства стовпчиків.

Bigtable – це велика і розподілена система для об’єктів, що синхронізуються, тому замість них застосовується розподілений сервіс блокувань (distributed lock service), який в Google називають Chubby. Його роль в Bigtable можна порівняти з роллю транзакцій в звичайних СУБД [27].

Для кожного таблет-сервера Chubby створює спеціальний chubby-файл. Завдяки цьому файлу Bigtable Master завжди відомо, які з серверів працездатні. Ще один chubby-файл містить посилання на розташування кореневого таблета (Root-tablet) з даними про розташування всіх інших таблетів. Цей файл повідомляє майстру, який з серверів якими таблетами керує.

Безумовно, використання сервісу Chubby в Bigtable якоюсь мірою вирішує завдання підтримки несуперечливості даних у розподіленому

середовищі з безліччю реплік. Але несуперечливість буває різною. Bigtable стала першою спробою досягти балансу між продуктивністю системи, її масштабованістю і несуперечливістю даних. Результатом стала підтримка так званої слабкої несуперечливості, яка, в принципі, задовольняє вимоги більшості працюючих з Bigtable сервісів.

На рис. 2.4 показано, як користувач шукає свій таблет [128].

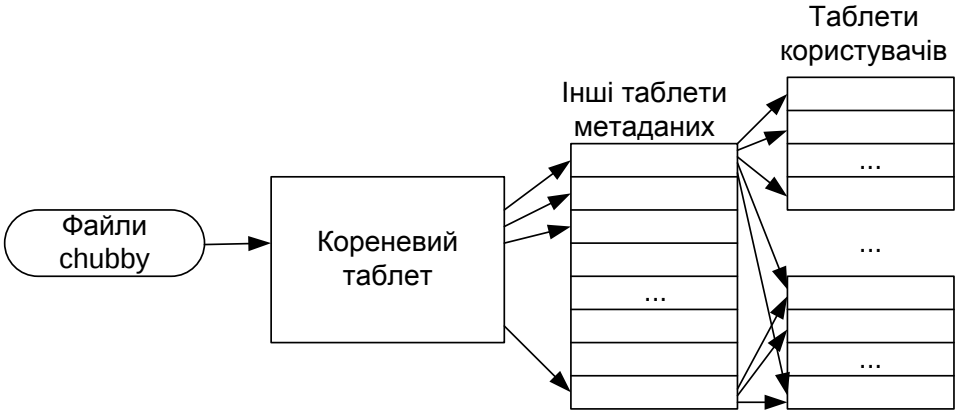


Рис. 2.4. Ієрархія таблетів

Носій моделі *«об'єкт-документ»* описується кортежами виду:

$$OD = \{< f_0, < f_1 : e_1,\ f_2 : e_2,\ f_n : e_n,\ f_{n+1} : d_1,\ f_2 : d_2,\ f_{n+l} : d_l >\}, \qquad (2.10)$$

де f_0 – ідентифікатор документу, $f_1..f_m$ – характеристики (атрибути) документу, $e_1..e_m$ – атомарні значення характристик $f_1..f_m$, $d_1..d_l$ – посилання на інші документи, $d_i = e(f_i)$.

Операції цієї моделі є об'єктні.

Операція визначення вузлів елемента

$$v(f_i) = \{C\} \cup \{f_{0_i} \mid i = \overline{1,n}\} \cup \{e(f_i) \mid i = \overline{0,n+l}\}, \qquad (2.11)$$

де C – колекція документів f_{0_i}.

Операція визначення значень вузлів:

$$v(f_i) = \{n_{e_j, f_i} \mid i = \overline{1,n},\ j = \overline{0,m+l}\}, \qquad (2.12)$$

де e_j – значення атрибутів f_i.

Також визначено відношення над елементами носія.

Відношення «елемент-елемент» визначаються між документами та колекцією:

$$OD \times C \to EE. \tag{2.13}$$

Відношення «елемент-атрибут»:

$$f_i \times OD \to EA. \tag{2.14}$$

Відношення «елемент-посилання»:

$$f_i \times d_j \to ER. \tag{2.15}$$

Відношення «елемент-дані» визначаються наступним чином:

$$f_i \times e_j \to ED. \tag{2.16}$$

Прикладами СУБД цього типу є MongoDB та CouchDB.

Графова модель даних подана як:

$$O = \langle ID, A, z, r \rangle, \tag{2.17}$$

де *ID* - множина ідентифікаторів, вузлів графа; *A* — множина позначених спрямованих дуг (*p,l,c*), $p, c \in ID$, *l* – «рядок-мітка», запис *(p,l,c)* означає, що між вузлами *p* та *c* є зв'язок *l;* z – функція, що відображає кожний вузол $n \in ID$ в конкретне значення складеного або атомарного типу, $z : n \to v$; *V* – особливий кореневий вузол графа.

Структура XML-документа, що складається з вкладених елементів-тегів добре відома, її відмінність від розглянутої вище графової моделі полягає, в основному, у трактуванні тегів і міток: в графах мітки використовуються як позначення зв'язків між елементами схем даних, і мітки не потрібні для позначення елемента, а в XML документно-орієнтованій моделі потрібно, щоб кожний (нетекстовий) елемент даних мав ідентифікуючу ознаку. Також XML транслюється в структуру даних «дерево», що є частковим випадком графової моделі.

У графовій моделі XML для слабоструктурованих даних необхідно

використовувати спеціалізовані типи атрибутів, такі як *ID*, *IDREF*, *IDREFS*. Зазначені типи дають змогу організувати зберігання перехресних посилань в *XML*-елементах виду <*eid,vahie*> (<ідентифікатор елемента, значення>) і атрибутах виду < *label, eid*> (<мітка, значення>).

Існує кілька видів *RDF*-даних як графової моделі: *RDF* / *XML, N3, Turtle, RDF* / *JSON*.

Опис ресурсів у вигляді *RDF*-набору даних – це трійка «суб'єкт»-«предикат»-«об'єкт», тобто для множини *U* (*Universal Resource Identifier, URI*, уніфікований ідентифікатор ресурсів) – елементи *f*, множини *B {Black nodes,* порожніх вузлів), множини *L {Literal, RDF*-літералів), $B \in e, L \in e$, визначається набір (*f, e*(*f*), e), де *f* – «суб'єкт»; *e*(*f*) – «предикат»; *e* – «об'єкт».

RDF-графова модель даних: нехай t = (*f, e(f), e*) є *RDF*-елементом даних, де $(f,e(f),e) \in (UB) \times U \times (UBL)$, причому *t* називається основним, якщо він не містить вузлів, які не мають ідентифікаторів. RDF-граф *G* є множиною $T \supseteq t$ [3].

Отже, Великі дані поєднують різні моделі даних. Для цього повинні існувати методи їх перетворення з мінімальною втратою даних.

Однією з технологій, що доцільно використовувати для роботи з Великими даними регіону, є простір даних.

2.4. Використання простору даних для моделювання Великих даних

Оскільки для формування асоціацій між сутностями та характеристиками необхідно працювати з різними джерелами даних, в яких один і той самий об'єкт може бути представлений під різними назвами, то для порівняння схем джерел даних доцільно використовувати простір даних з каталогом даних та словником даних для порівняння назв об'єктів.

Простір даних – це блоковий вектор, що містить множину інформаційних продуктів предметної області, поділену на три блоки: структуровані дані (бази даних, сховища даних), напівструктуровані дані (XML, електронні таблиці) та неструктуровані дані (текст). Над цим вектором та його окремими елементами визначено операції та предикати, які забезпечують [3]:

- перетворення різних елементів вектора один в одного;
- об’єднання елементів одного типу;
- пошук в елементах за ключовим словом.

Моделі даних джерел, що можуть бути частиною Великих даних, утворюють ієрархію відповідно до їх виразної потужності [31]:

- реляційна модель,
- багатовимірна модель,
- об’єктно-реляційна модель,
- об’єктно-документна модель,
- розширена мова розмітки інформації (XML) зі схемою,
- середовище описання ресурсів (Resource Description Framework – RDF),
- стандартний засіб описання зв’язків між об’єктами даних – онтології, описані за допомогою Web Ontology Language – OWL,
- структурований текст (у тому числі HTML, Excel),
- слабоструктурований текст.

Придатність моделей даних до підтримання мов запитів та до використання в глобальній мережі подано на рис. 2.5 [30].

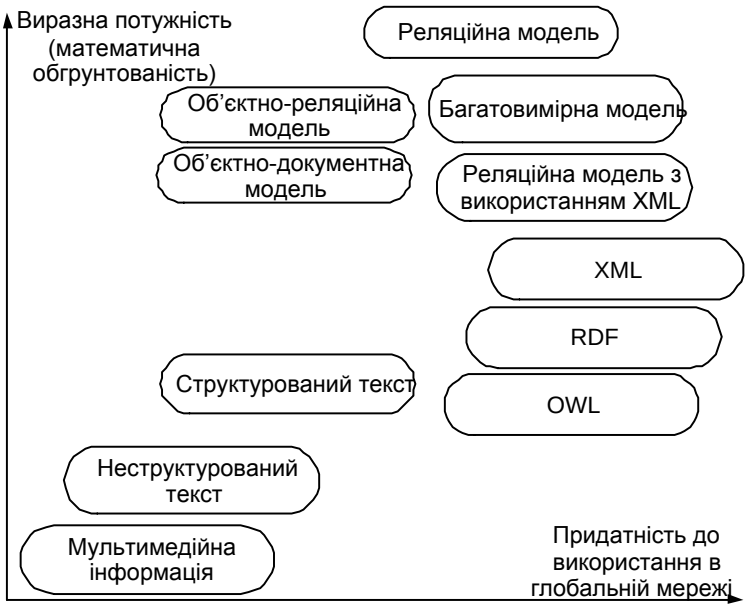


Рис. 2.5. Придатність моделей даних до підтримання мов запитів та до використання в глобальній мережі

Кожен учасник простору даних підтримує деяку модель даних і деяку мову запитів, відповідну цій моделі. Запит до такого програмного засобу відповідає тому, що зазвичай підтримується у файлових системах стосовно до їх директорій: зіставлення імен, пошук в діапазоні дат, сортування за розміром файлу та ін. На наступному рівні простору даних модель даних повинна підтримувати мультимножини слів з метою здійснення ефективного пошуку необхідної інформації за ключовими словами. Нижче рівня моделі мультимножини слів в ієрархії може розташовуватися модель напівструктурованих даних, заснована на позначених графах. Оскільки джерела даних є різнотипні, то необхідно визначити платформу та архітектуру ПД.

Платформа підтримання ПД (ПППД) – це набір програмного забезпечення, що керує організацією, зберіганням і пошуком даних у просторі. Також здійснює контроль безпеки та цілісності.

Архітектуру простору даних спроектовано за рівнями (рис. 2.6). Рівень застосувань призначений для реалізації операцій над даними у просторі даних. Рівень онтологій використовується для встановлення зв'язку між джерелами.

Останній рівень містить джерела даних та забезпечує доступ до даних та виконання операцій рівня застосувань безпосередньо в джерелі (наприклад, операція вибірки на рівні реалізації виконується як запит в конкретній базі даних).

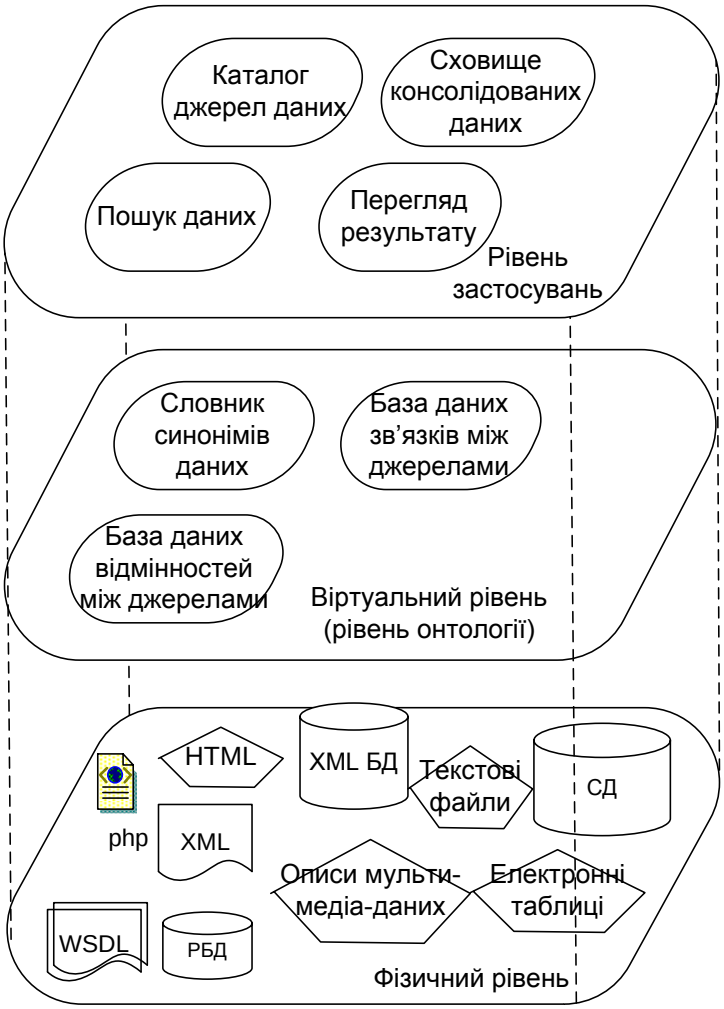


Рис. 2.6. Рівні реалізації фізичної моделі простору даних

Формалізація каталогу простору даних

Каталог *Cg* – це реєстр ресурсів даних, що містить базову інформацію про кожного з них: джерело, ім'я, місцезнаходження в джерелі, розмір, дату створення і власника, технологічну платформу, протоколи та режим доступу, частоту оновлення та ін. Формується на основі метаописань джерел даних:

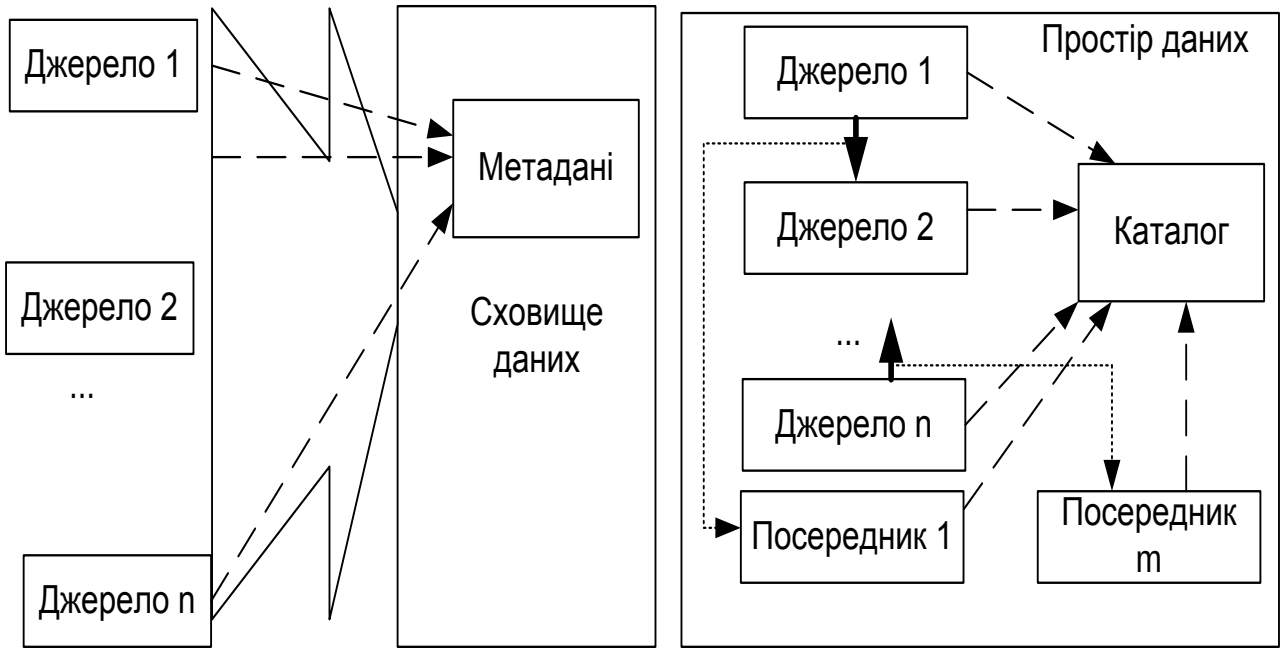
$$\text{Metadata}(\mathbf{DB}, \mathbf{DW}, \mathbf{Wb}, \mathbf{Nd}, \mathbf{Gr}) \Rightarrow Cg. \qquad (2.18)$$

Він не лише містить описову інформацію (тобто виконує роль

метаданих), але й зберігає для кожного учасника схему джерела, статистичні дані, швидкість зміни, точність, можливості відповідей на запити, інформацію про власника і дані, про політику доступу і підтримання конфіденційності.

Оскільки джерела простору даних фізично не переносять у нього інформацію та не можуть обмінюватись між собою інформацією, то у каталозі необхідно зберігати дані і про зв'язки між джерелами даних.

Зазначимо, що поняття каталогу простору даних ширше, ніж поняття метаданих простору даних. Окрім традиційних за Захманом [30] 6-ти типів метаданих, каталог простору даних зберігає ще інформацію про зв'язки між джерелами і протоколи обміну інформацією між ними. Відмінності між поданням джерел даних у метаданих та каталозі схематично подані на рис.2.7 [30].



а) б)

Рис. 2.7. Подання даних у: а) метаданих у сховищі даних; б) каталозі простору даних.

Зв’язки у каталозі можуть зберігатися у вигляді:

- метаданих;
- перетворень запитів;
- графів залежності;
- текстових описань тощо.

Над каталогом розміщене середовище керування моделями **EM**, яке дає змогу створювати нові зв’язки і маніпулювати наявними зв’язками (наприклад, об’єднувати або інвертувати відображення, зливати схеми і створювати єдині подання декількох джерел).

Важливою компонентою простору даних є федеративне сховище даних (Cg'), яке слугує для досягнення наступних цілей:

- створення асоціацій між об’єктами даних від різних учасників;
- вдосконалення доступу до джерел з обмеженими власними засобами доступу;
- забезпечення можливості виконання деяких запитів без доступу до реального джерела даних;
- консолідації даних як результату запиту користувача;
- підтримання високого рівня доступності і відновлення.

Отже, зв’язок між каталогом Cg, середовищем керування моделями **EM** і федералізованим сховищем даних Cg' можна подати як відображення:

$$\mathbf{EM}(Cg) \Rightarrow Cg'. \qquad (2.19)$$

Що більше моделей здатне «розрізнити» середовище керування, то точнішою буде інформація в Cg' і ефективніше можна буде здійснювати процедури інтеграції, пошуку та опрацювання даних у просторі даних **DS.**

Зв’язок між елементами простору даних поданий на рис. 2.8.

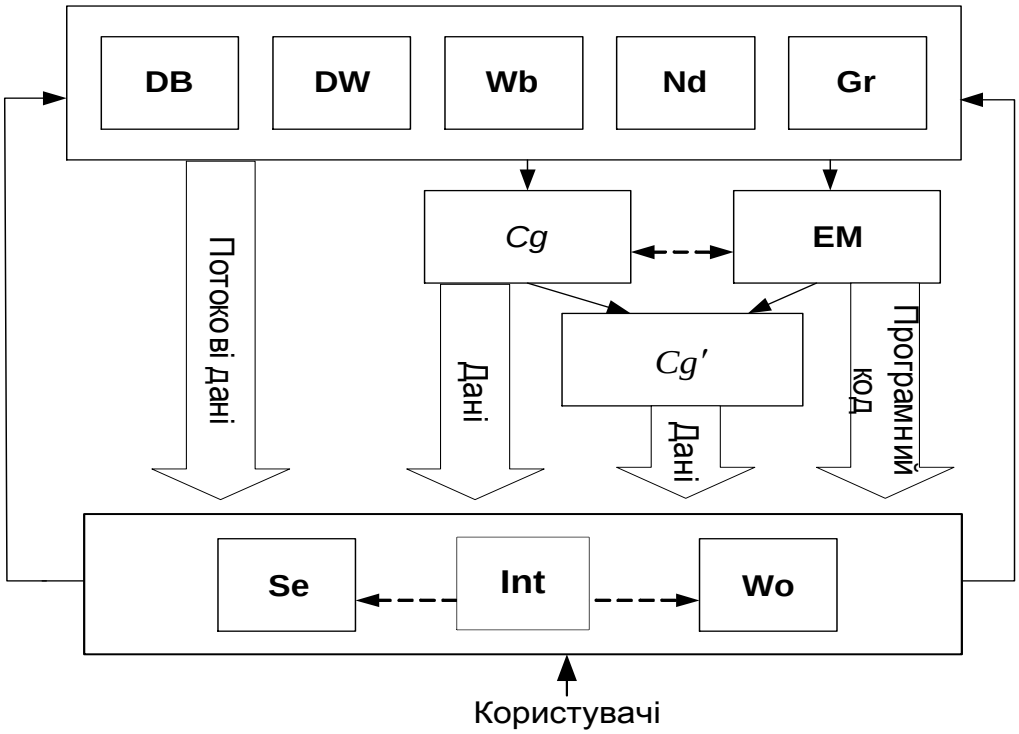


Рис. 2.8. Схема елементів простору даних

2.5. Модель федеративного сховища Великих даних

Для технології Великих даних необхідним є опрацювання інформації з різних за виразною потужністю типів джерел інформації: структурованих, слабоструктурованих, неструктурованих. Відповідно федеративне сховище даних, побудоване на їх основі, містить:

1) реляційні бази даних,
2) багатовимірні бази даних,
3) бази даних XML,
4) бази даних NoSQL,
5) файлове сховище,
6) репозиторій метаданих,
7) інтегратор джерел даних,
8) подання для доступу до сховища (рис. 2.9).

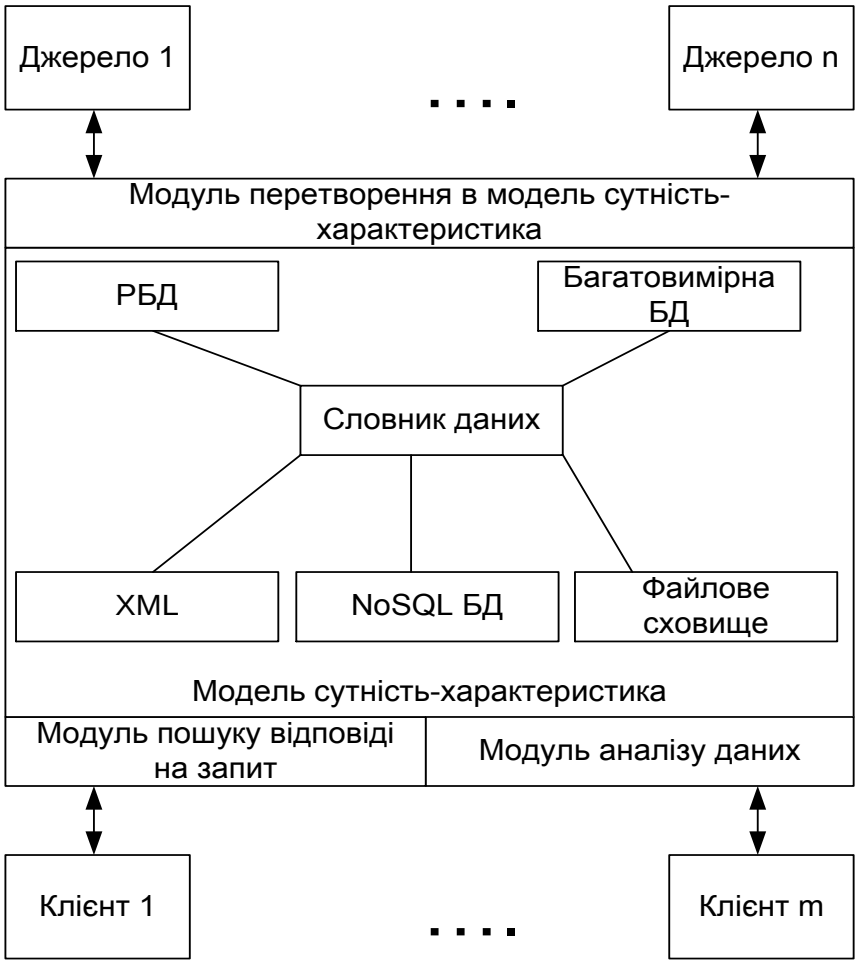


Рис. 2.9. Архітектура федеративного сховища даних

Отже, базовим елементом федеративного сховища даних для роботи з Великими даними є модуль перетворення інших моделей даних в модель «сутність-характеристика». З допомогою словника даних, організованого на основі каталогу простору даних, здійснюється визначення синонімів спільних понять сутностей чи характеристик.

Після он-лайн перетворення даних в модель «сутність-характеристика» уможливлюється пошук відповіді на запит користувача чи аналіз даних.

Основними характерними властивостями, які відрізняють федеративні сховища даних для Великих даних від інших сховищ даних, є наступні:

1) Наявність своєї системи керування сховищем даних, за допомогою якої здійснюється робота із сховищем (виконання запитів до сховища).

2) Наявність у сховищі реляційної БД, основним призначенням якої є зберігання структурованих даних і даних, до яких здійснюється частий доступ.
3) Наявність у сховищі багатовимірної БД, яка може містити як атомарні, так і узагальнені дані. Основним призначенням багатовимірної бази даних є зберігання даних, до яких виконуються складні запити.
4) Наявність у сховищі бази даних XML та баз даних NoSQL, основним призначенням яких є зберігання слабоструктурованих даних і слабоструктурованої частини частково-структурованих даних;
5) Збереження неструктурованих даних у вигляді файлів, що зберігаються безпосередньо у файловій системі.
6) Взаємодія з джерелами даних, що здійснюється за допомогою інтегратора, та полягає у відслідковуванні змін даних і метаданих, які відбуваються у джерелах, і застосуванні цих змін відповідно до налаштувань сховища даних.
7) Уніфікований доступ користувачів до сховища даних, який дає змогу користувачам звертатися до даних за допомогою єдиного інтерфейсу, незалежно від фізичного та логічного розташування цих даних у сховищі.

Необхідність наявності баз даних NoSQL виникає через:

1) проблеми розширення РБД, коли набір даних занадто великий;
2) СУБД не були призначені для дистрибуції;
3) поширення бази даних багатовузлових рішень;
4) необхідність у вертикальному масштабуванні або горизонтальному масштабуванні.

Іншими шляхами скалювання РБД є:

1) Мульти-майстер реплікації – система реплікації з декількома майстрами відповідає за поширення зміни даних, які зроблені кожним учасником в решті частини групи, і вирішення конфліктів, які можуть

виникнути через одночасні зміни, зроблені різними членами.

2) Дані тільки додаються, а не оновлюються та знищуються.
3) Зменшення використання операцій з'єднання (Join), тим самим зменшення часу запиту (включаючи в тому числі денормалізовані дані, які призводять до появи ще більших баз даних).
4) Бази даних в пам'яті (in-memory).

Серед причин появи NoSQL треба виділити:

1) вибух соціальних мереж (Facebook, Twitter) з великими потребами в даних;
2) поява хмарних рішень, таких як Amazon S3 (рішення простого зберігання);
3) використання динамічно типізованих мов (Ruby/Groovy), зрушення в динамічно-типізованих даних з частими змінами схеми;
4) формування спільнот, що використовують та розробляють програмне забезпечення з відкритим вихідним кодом.

При проектуванні федеративного сховища Великих даних для забезпечення його оптимального функціонування пропонується поєднати підходи проектування на базі структурованості джерел даних і запитів до сховища даних. В основі цього лежить CAP-теорема [15]. За цією теоремою є три властивості системи: узгодженість, доступність і подільність. Можна мати не більше двох з цих трьох властивостей для будь-якої системи з розділюваними даними. Проте, щоб масштабувати, необхідно розбити на розділи, що у свою чергу, призводить до втрати узгодженості та доступності.

Нехай N – кількість вузлів з реплікою даних, W – кількість вузлів, які мають підтвердити оновлення, R – мінімальна кількість вузлів, які встигають здійснювати операцію читання. Тоді для $W + R>N$ виконується вимога суворої узгодженості [27].

Для задоволення вимог CAP-теореми спроектуємо інформаційну структуру Великих даних (рис. 2.10).

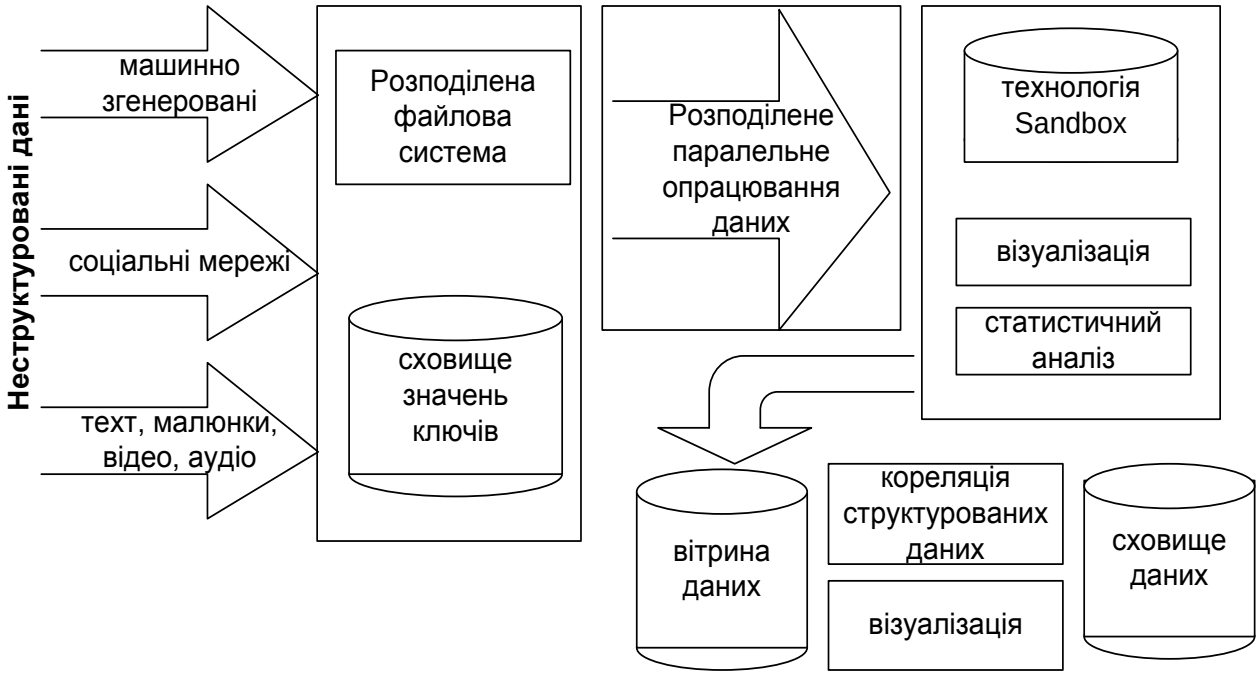


Рис. 2.10. Інформаційна структура Великих даних

Для роботи з Великими даними передбачається чотири фази перетворення даних:

1) Набуття – робота з фіксованими або змінними даними з використанням розподілених файлових систем, даними (Hadoop Distributed File Systems [HDFSs]) і базами даних NoSQL (Oracle NoSQL баз даних).
2) Організація - парадигма програмування MapReduce використовується для інтерпретації й уточнення даних.
3) Аналіз - очищені і організовані дані подаються в реляційну базу даних (бази даних SQL), щоб здійснити належний аналіз.
4) Підтримка рішень – використання методів підтримки прийняття рішень для подальшого аналізу даних.

Ці чотири фази обробки даних не можуть здійснюватись з використанням однієї машини.

Вважається, що Великі дані є технологією Hadoop. Проте Великі дані є

дещо більше, ніж Hadoop. Однією з ключових вимог є розуміння і навігація по федеративних джерелах Великих даних для виявлення даних. Нова технологія підтримує також індекси, пошук та навігацію різних джерел Великих даних. Hadoop являє собою набір можливостей з відкритим вихідним кодом. Два з найбільш відомих з них є [28]: Hadoop FS для зберігання різноманітної інформації, MapReduce - двигун паралельної обробки. Сховища даних також керують Великими даними, оскільки обсяг структурованих даних швидко зростає. Можливість запуску глибоких аналітичних запитів на величезних обсягах структурованих даних є великою проблемою даних. Це вимагає наявності величезних сховищ даних паралельної обробки і спеціально побудованих засобів для аналізу даних. Великі дані містять не тільки статичні, але й динамічні дані. Потокові дані представляють абсолютно іншу проблему Великих даних – здатність швидко аналізувати і опрацьовувати дані в той час як вони ще прибувають.

Висновки до розділу 2

1. У розділі побудовано модель даних «сутність-характеристика» для Великих даних, що дає змогу опрацьовувати дані різних форматів.
2. Для аналітичного опрацювання Великих даних використано федеративне сховище даних, що дає змогу працювати з Великими даними без проміжного збереження і тим самим зменшує обчислювальну та ємнісну складність виконання запитів.
3. Визначено моделі асоціацій об’єктів та характеристик основних представлень даних в NoSQL. Побудовано інформаційну структуру Великих даних. Усе це стало основою для продовження досліджень і зосередженню на проблемі опрацювання різнорідних даних без їх попередньої інтеграції.

4. Для представлення Великих даних використано простір даних, який дає змогу працювати з різнотипними даними. Проте основною операцією інтеграції даних є не консолідація, а федералізація, що дає змогу зменшувати ємнісну складність запитів.

   Матеріали розділу опубліковано у [1–3, 9, 7, 11, 13].

РОЗДІЛ 3. РОЗРОБЛЕННЯ МЕТОДУ АНАЛІЗУ ДАНИХ В МОДЕЛІ «СУТНІСТЬ-ХАРАКТЕРИСТИКА»

У розділі розроблено метод обміну різнотипними даними та приведення цих типів даних до моделі «сутність-характеристика». Розроблено метод відповіді на запит користувача з врахуванням типу даних джерела.

3.1. Метод перетворення даних в модель «сутність-характеристика»

Основною метою перетворення даних в модель «сутність-характеристика» є забезпечення можливості опрацювання даних будь-якої структури.

Метод перетворення даних в модель сутність характеристика полягає у перерахунку важливості характеристики для сутності, а також фізичному перетворенні схеми даних у пару «сутність-характеристика».

Схему перетворення даних в модель «сутність-характеристика» подано на рис. 3.1.

Для розширення *XML* пропонується використовувати архітектуру *MVVM* («*View-Model*», «Вигляд-Модель»). Цей шаблон застосовується при проектуванні архітектури програмного додатка й використовує поділ моделі в частині логіки її функціонування і її подання на користувацькому інтерфейсі. Основною її відмінністю від відомої архітектури програмного додатка *MCV* (*Model-View-Controller,* Модель-Вигляд-Контролер) є відсутність вимоги прив’язки даних до їхнього подання.

Відповідно до цього підходу покажемо необхідні розроблені компоненти для автоматизованої конвертації з реляційної БД в модель XML з врахуванням особливостей моделі «сутність-характеристика» на рис. 3.2.

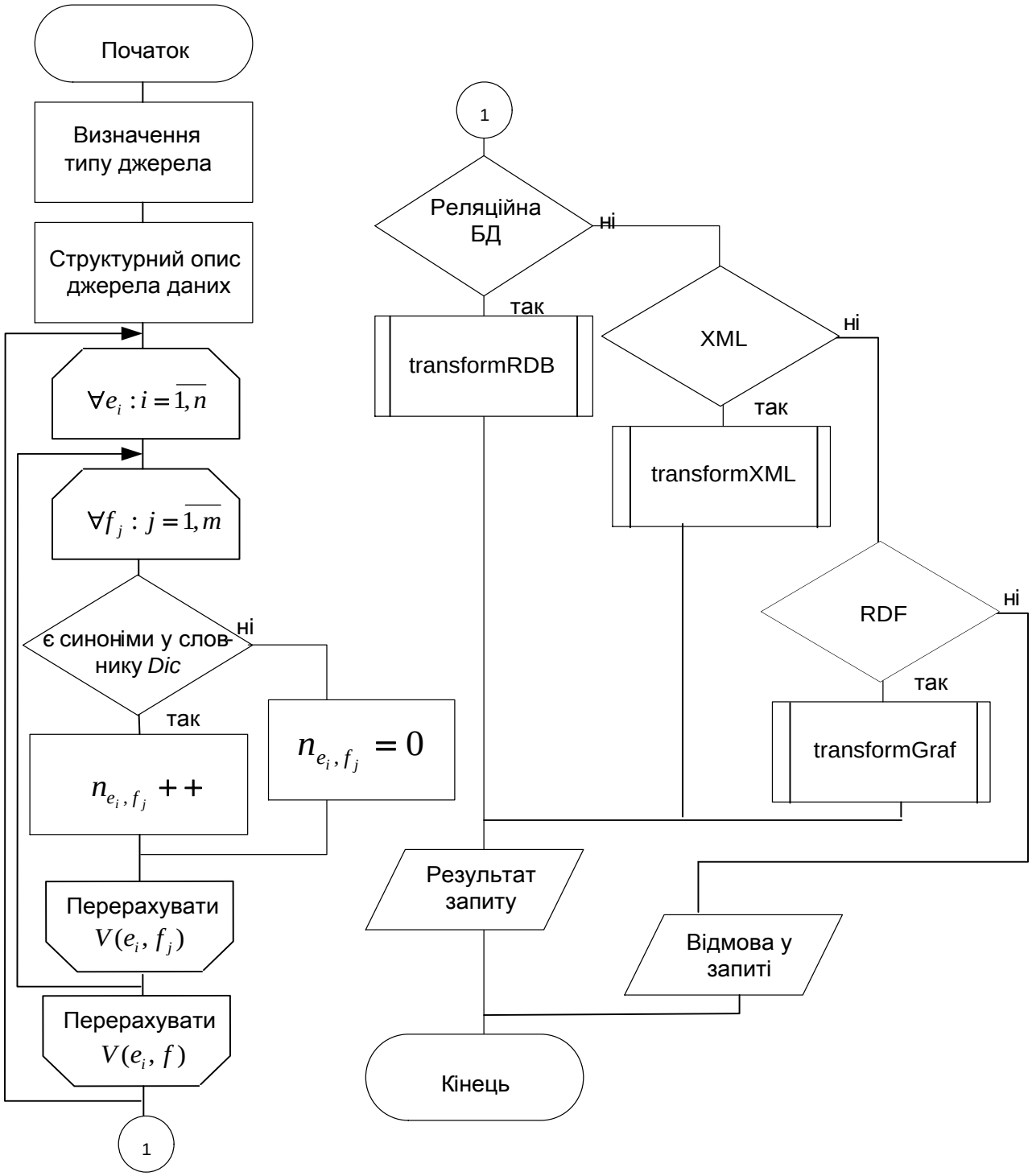


Рис. 3.1. Схема перетворення даних в модель «сутність-характеристика»

В архітектурі додатків, які реалізуються за структурою, яку показано на рис. 3.2, присутні кілька компонентів, які вимагають додаткових пояснень. Один з таких компонентів – *XML*-розмітка. Вказана розмітка також може бути використана для роботи з документно-орієнтованими базами даних. Тому

перетворення з моделі «об’єкт-документ» в модель «сутність-харакетристика» не виконувалось.

Рис. 3.2. Схема конвертації XML-БД з врахуванням моделі «сутність-характеристика»

Наведемо приклад *XML* опису вимірів системи туристичного комплексу.

<measurement>
<name> Станція </name>
<location> Красне </ location>
<time> 2014-04-14 09:41</ time>
</measurement>

```
<brake>
<fuzzy name="brakel">
<function minvalue="0"> 0 </ function>
Cfunction minvalue ="1" maxvalue="5"> x/(maxvalue
-Minvalue) + maxvalue/(maxvalue-minvalue) </ function>
</ fuzzy>
<fuzzy name="brake2">
<function minvalue="6"> 6 </ function>
<function minvalue="7" maxvalue="9">x/(maxvalue
-minvalue) - minvalue/(maxvalue-minvalue)</function>
</ fuzzy>
</ brake>
</measurement>
```

Для зручності універсального опису елементів можна застосовувати Xsd опис, вид якого подано нижче. Функція приналежності в даному прикладі обмежена значеннями Minvalue і Maxvalue.

```
<>s: schema id="Fuzzyshema " elementformdefault="F qualified" xmlns
    :xs="http://www.w3.org/2001/XMLSchttp">

<xs:complextypename="fuzzy">
<xs:sequence>
    <xs:elementname="fuzzy"minoccurs="1maxoccurs="un-bounded"
    type="functions7>
</xs:sequence>
</xs:complextype>
<xs:complextypename="functions">
<xs:sequence>
<xs:elementref="function"minoccurs="1"maxoccurs="unbounded"/>
</xs:sequence>
<xs:attributename="name"type="xs:string"/>
</xs:complextype>
<xs:elementname="function">
<xs:complextype>
<xs:simplecontent>
<xs:extensionbase="xs:string">
<xs:attributename="minvalue"type="xs:string"/>
<xs:attributename="maxvalue"type="xs:string"/>
</xs:extension>
</xs:simplecontent>
</xs:complextype>
</:ts:element>
</xs:schema>
```

У реальних умовах може знадобитися фільтрування даних. На рис. 3.3 ілюструється фільтрація для джерела даних у вигляді XML-файлу з ADM.

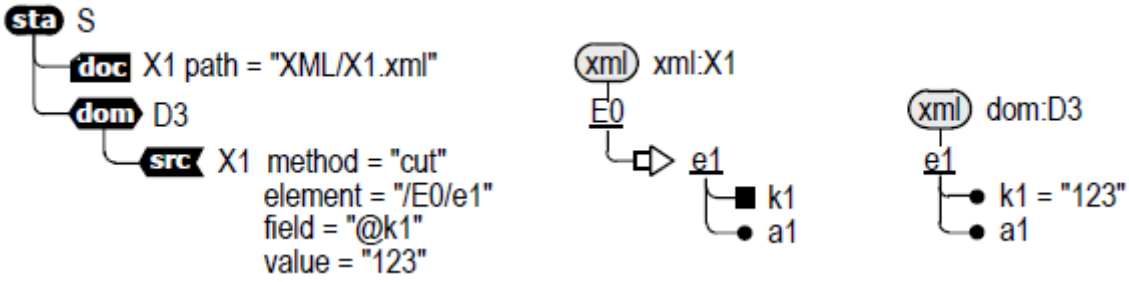


Рис. 3.3. Приклад завантаження Dom-об’єкта з фільтрацією XML-даних: а) фрагмент динамічної моделі; б) модель XML-даних джерела; в) модель XML-даних, що завантажуються в Dom-об’єкт.

У динамічній моделі на рис. 3.3,а в кореневому стані sta:S задані визначення XML-документа doc:X1, що зберігається в ADM, і Dom-елемента dom:D3, у якому передбачено завантаження документа doc:X1 з фільтрацією. Джерело даних src:X1 має атрибут method = «cut», що вказує на те, що з вихідного Xml-документа буде «вирізане» піддерево. Додаткові атрибути «element», «field» і «value», задають умову фільтрації. Атрибут «element» містить Xpath-вираз, який визначає множину вузлів-елементів в XML-дереві, один з яких буде коренем завантажених піддерев. Атрибут «field» містить Xpath-вираз, який задає в піддереві, що перевіряється XML-елемент або атрибут. Атрибут «value» містить необхідне значення. У поданому прикладі потрібно завантажити в Dom-об’єкт піддерево, що починається в елементі «e1», атрибут «k1» якого має значення «123».

На рис. 3.3,б представлена модель XML-даних doc:X1 з кореневим XML-елементом «E0», він може містити дочірні XML-елементи «e1», кожний з яких містить атрибути «k1» (ідентифікатор) і «a1». Таким чином, Xpath-вираз «/E0/e1», заданий в атрибуті «element» джерела src:X1 на рис. 3.3,а, адресує множину усіх XML-елементів «e1»; атрибут «field» адресує в «e1» XML-атрибут «k1», а атрибут «value» задає для нього шукане значення «123».

Під час опрацювання елемента dom:D3 інтерпретатор звертається до джерела src:X1, перебирає XML-елементи «e1», відшукуючи той, у якого XML-атрибут «k1» має шукане значення «123», і завантажує відповідне піддерево в

Dom-об’єкт. У результаті в Dom-об’єкт «D3» будуть завантажені XML-дані, що відповідають моделі, представленій на рис. 3.3,в.

Далі розроблено метод перетворення реляційних даних у модель «сутність- характеристика» transformRDB, який містить наступні кроки.

Крок 1. Створити кореневий елемент схеми *RDB* для моделі даних *Big Data.*

Крок 2. Для кожної сутності *e* створити окремий елемент Entity і розмістити його під кореневим елементом.

Крок 3. Для кожної характеристики *f* сутності *e* з важливістю більшою за порогове значення створити атрибут Feature, розмістити його усередині відповідного опису сутності та задати його тип.

Крок 4. В атрибуті необхідно вибрати й задати тип даних та визначити граничні значення.

Метод перетворення з формату JSON нагадує структуру методу перетворення з формату XML і тому окремо не розглядається.

Для перетворення графової моделі в модель «сутність-харакетристика» важливим є визначення ваги зв’язку між елементами. Оскільки першим параметром моделі є характеристика, другим – зв’язок, а третім – сутність, то перетворення між моделями полягатиме у числовому вираженні асоціації між елементами RDF-моделі (рис. 3.4).

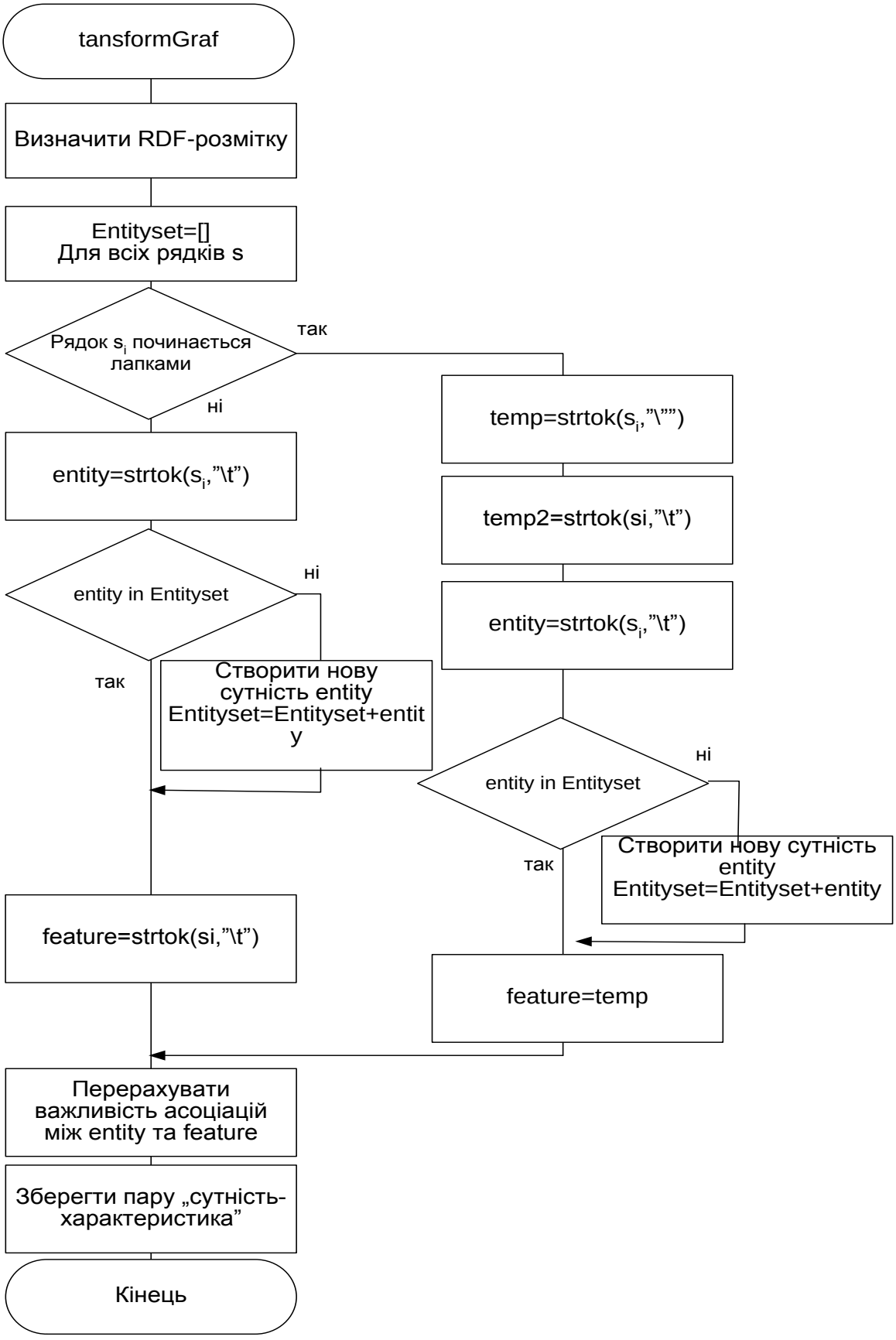


Рис. 3.4. Схема конвертації графової БД з врахуванням моделі «сутність-характеристика»

Алгоритм потокової обробки елементів документа джерела починає роботу з ініціалізації інструмента потокового введення Xmlreader. Далі в циклі виконується потокове читання вузла за вузлом з документа джерела. Перевіряється, чи є присутнім в елементі-джерелі атрибут wanted, і в цьому випадку перевіряється, чи оброблюваний вузол є шуканим, у циклі перебираються імена зі списку, заданого в атрибуті wanted, порівняються з іменем оброблюваного вузла й у випадку співпадіння встановлюється прапорець.

По завершенню циклу перевіряється прапорець збігу і, якщо прапорець не встановлений, завершується обробка даного вузла й виконується перехід до обробки наступного. Якщо оброблюваний вузол виявився шуканим, він перевіряється на виконання умов фільтрації, які задані в атрибуті cond. Такі умови вимагають звернення до інших вузлів документа з потоку введення. Для цього оброблюваний елемент (разом з вмістом) видобувається в кеш.

Далі в циклі з атрибута cond отримуються умови й перевіряються для оброблюваного вузла, що перебуває в кеші. Якщо оброблюваний вузол пройшов усі перевірки, він приєднується як дочірній елемент до оброблюваного цільового елемента батьківського документа.

Далі виконується очищення приєднаного дочірнього елемента від внутрішніх елементів, імена яких задані в атрибуті ignory. Після того як оброблюваний вузол документа джерела перевірений, приєднаний до цільового елемента й очищений, для нього рекурсивно проробляються внутрішні джерела даних (рис. 3.5).

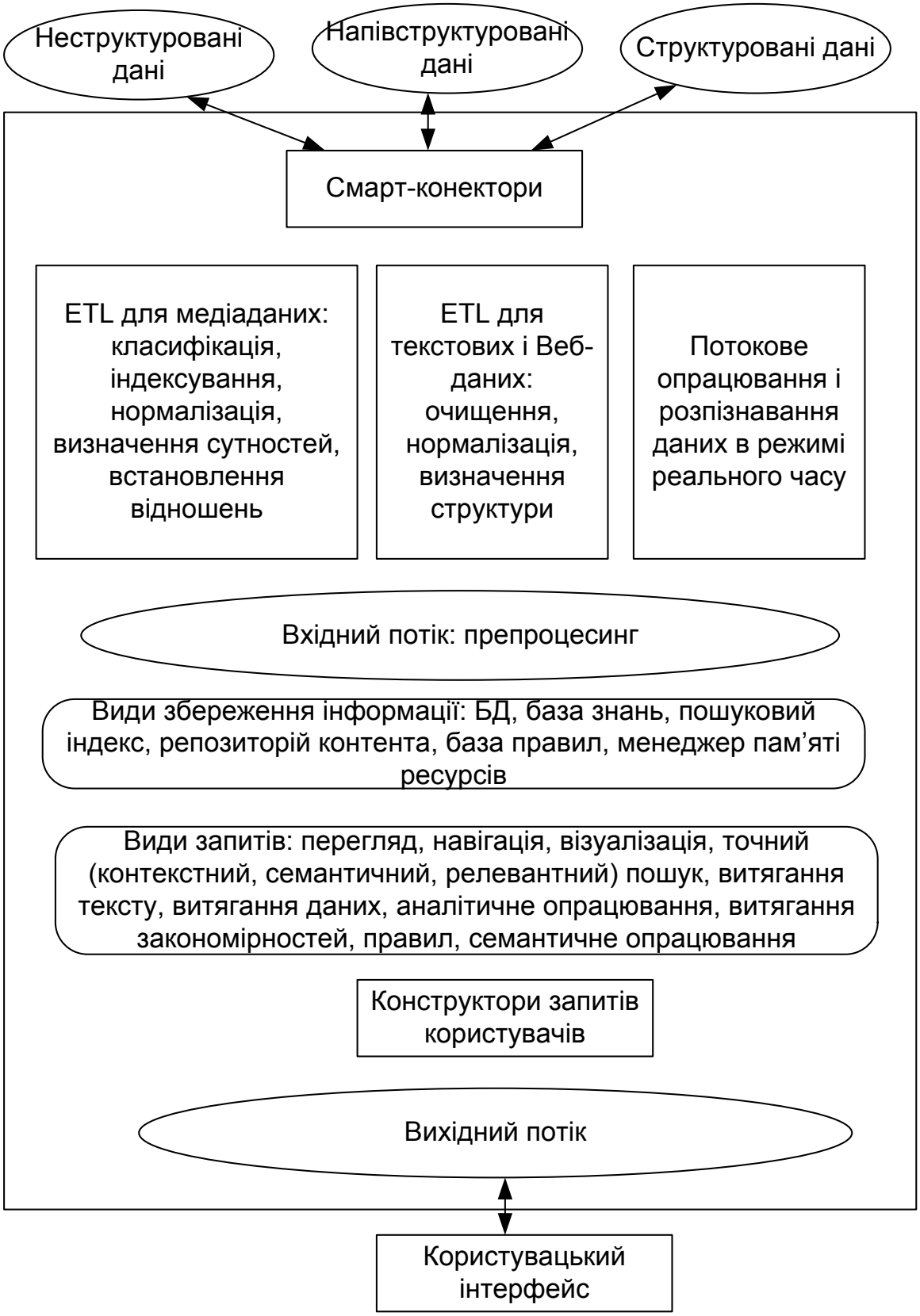


Рис. 3.5. Універсальна структура інтелектуального аналізу мультиструктурної інформації

3.2. Розроблення методу прогнозування процесів розвитку регіону

Задачею прогнозування є знаходження залежності між датою та параметрами ресурсів (оздоровчий, історичний, культурний, відпочинковий), отриманих з бази даних вказаної структури, на основі аналізу попередньо отриманих даних. Динаміка зміни виробленої потужності може бути охарактеризована стосовно якогось базисного (зазвичай першого) спостереження і величиною зміни сусідніх рівнів. У якості статистичних характеристик часового ряду Y_i, $i = 1..n$ використовуються наступні величини: середнє арифметичне $Y = \frac{1}{N}\sum_{j=1}^{N} Y_j$, середній абсолютний приріст $\overline{Y} = (Y_n - Y_1)/(N-1)$, де N – число рівнів ряду, Y_i – рівні ряду

Відповідно до методу перевірки істотності різниці середніх початковий часовий ряд розбивається на дві однакові (або майже однакові) частини, після чого перевіряється гіпотеза про істотність різниці середніх для цих частин. Недолік методу полягає в неможливості правильно визначити наявність тренда у тому випадку, коли часовий ряд містить точку зміни тенденції в районі середини ряду.

У методі Форстера-Стюарта гіпотеза про відсутність тренда перевіряється за допомогою допоміжних функцій [19]:

$$L = \sum_{t=2}^{N} I_t,\ I_t = U_t - V_t,\ u_t = \begin{cases} 1, Y_t < Y_{t-1},...,Y_1 \\ 0 \end{cases},\ v_t = \begin{cases} 1, Y_t > Y_{t-1},...,Y_1 \\ 0 \end{cases}. \qquad (3.1)$$

Гіпотеза про відсутність тенденції не приймається, якщо розрахункове t-значення більше табличного на вибраному рівні значущості 0.95.

Перевірка однорідності даних здійснюється на основі критерію Ірвіна, що заснований на порівнянні сусідніх значень ряду. Відповідно до нього розраховується характеристика t: $t = \frac{Y_t - Y_{t-1}}{Y}$.

Отримані значення порівнюються з табличними значеннями. Проте критерій Ірвіна недостатньо ефективний для виявлення аномальності в динамічних рядах, тому що величина ***y*** характеризує відхилення значень показника від середнього рівня по всій сукупності спостережень, а значить, він не виявляє викиди усередині ряду спостережень. У дисертації використовується модифікований метод, відповідно до якого послідовно розраховують ***y*** не по всій сукупності, а по 3-4 спостереженням, і розраховані з такими ковзаючими значеннями **y** величини порівнюються з критичними значеннями для ***n=3***.

Оцінка властивостей зводиться до дослідження автокореляційної функції вихідного і різницевих рядів. Аналіз автокореляції виконується за допомогою графіка і критичних значень коефіцієнтів, встановлених експертно.

Проте, якщо аналізувати просторові дані, виникає ряд нюансів. У просторових даних є дві властивості, які ускладнюють вимоги для проведення статистичного аналізу (наприклад, МНК) порівняно із звичайними (непросторовими) даними. Втім, ці властивості і не роблять такий аналіз неможливим [22].

По-перше, географічні дані набагато частіше, ніж непросторові дані, вже автокорельовані. Це означає, що об’єкти, розташовані в безпосередній близькості один від одного, зазвичай більше схожі, ніж об’єкти, розташовані у віддаленні один від одного. При використанні звичайних (непросторових) методів відбувається зайвий підрахунок (переоблік) ступеня впливу цього фактора.

По-друге, для просторових даних важлива саме їх географія.

Часто найважливішими для моделі є непостійні процеси; ці процеси відбуваються по-різному на різних ділянках. Ці особливості можуть бути описані як територіальні зміни або просторовий дрейф.

Тому розроблені спеціальні методи регресійного аналізу, які акцентовані саме на ці дві особливості просторових даних. Вони призначені для поліпшення моделювання зв’язку саме таких даних. Одні просторові регресійні методи

найефективніше враховують просторову автокореляцію, інші – географічну непостійність явищ. Сьогодні не існує просторових регресійних методів, які були б ефективні відразу для всіх особливостей просторових даних.

На перший погляд може здатися, що ставлення до просторової автокореляції у людини, яка працює з просторовими даними, і у людини, яка використовує непросторові дані, має суттєво відрізнятися. При використанні непросторових даних автокорреляція здається марною функцією, яка не покращує результат кореляції а, навпаки, порушує основні принципи аналізу, які традиційно використовуються при роботі з непросторовими даними. Доводиться виключати її вплив (наприклад, через перевиборку об’єктів).

Для географа або ГІС-аналітика просторова автокореляція є прямим доказом важливості просторових процесів в роботі. Така інформація є узагальнюючою для всіх інших даних.

Просторові процеси і взаємини – основна причина, через яку географи так зацікавлені в просторовому аналізі даних.

Але для того, щоб уникнути великої обчислювальної складності процесу підрахунку впливу просторових чинників у моделі, треба визначити повний набір змінних, правильно описати всю структуру даних, від яких залежить об’єкт дослідження. Якщо ви не зможете знайти всі ці змінні, то найімовірніше будете спостерігати статистично істотний залишок автокореляції у своїй моделі. Поки цю помилку не виправлено, не можна буде довіряти отриманим результатам. Використовується інструмент просторової автокореляції (Spatial Autocorrelation tool) в наборі інструментів Просторова статистика (Spatial Statistic Toolbox) для перевірки результатів на наявність статистично істотного залишку автокореляції у моделі.

Існує, принаймні, три методи роботи з просторовою автокореляцією: зміна вихідних змінних, поділ просторових і непросторових змінних, впровадження просторової автокореляції в регресійну модель. Коротко розглянемо кожен з них [23].

Зміна вихідних змінних призводить до того, що просторова автокореляція виключається з вихідних змінних. У той же час це зовсім не гарантує того, що її немає взагалі. Набагато більш імовірно, що автокореляція продовжує бути присутньою для залежних змінних, адже позбутися необхідно саме від автокореляції для цих змінних. Цей спосіб підходить для людей, що працюють зі стандартними (непросторовими) даними. У цьому випадку (для непросторових даних) причиною появи автокореляції може бути надмірність даних (тобто здійснена вибірка зроблена занадто детально).

Поділ просторових і непросторових компонентів для кожної із змінних може проводитися шляхом використання регресійного методу просторового фільтрування. При цьому просторова складова видаляється у кожної з вихідних змінних, але додається назад в модель в якості нової змінної.

Включення просторової автокореляції в регресійну модель реалізується за допомогою використання просторових економетричних методів.

Є три поширених підходи до просторового аналізу. Перший полягає в тому, щоб спрощувати дані карти. Другий підхід полягає у з'ясуванні факту випадковості: коли аналіз карти полягає в тому, щоб довести, що або спостережуване явище являє собою закономірність, або перед нами випадкові факти. Нарешті, третій варіант пов'язаний з пошуком причинно-наслідкових зв'язків одних чинників з іншими.

3.3. Розроблення методу пошуку закономірностей

3.3.1. Множинна лінійна регресія

Множинна лінійна регресія (МЛР) [19] є підходом для моделювання взаємозв'язків між результативною змінною *y* та двома і більше факторними змінними *x*. МЛР має багато практичних застосувань, одним з них є прогнозування. В загальному МЛР має такий вигляд:

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2 + \varepsilon . \qquad (3.2)$$

Тут результативна змінна *y* моделюється як комбінація константних, лінійних, добуткових і квадратичних елементів, сформованих двома факторними ознаками x_1 та x_2. Результат МЛР є набір оцінених регресійних коефіцієнтів $a_i, (i = 1,...,5)$. Зазвичай коефіцієнти лінійної регресії знаходять методом найменших квадратів (МНК): модель застосовується так, що сума квадратів різниць спостережуваних і прогнозованих значень зводиться до мінімуму.

Моделі МЛР:

1. Лінійна — складається з константних та лінійних елементів:
$$Y = a_0 + \sum_{i=1}^{N} a_i x_i \,. \quad (3.3)$$
2. Добуткова — складається з константних, лінійних та добуткових елементів:
$$Y = a_0 + \sum_{i=1}^{N} a_i x_i + \sum_{i \neq j}^{N} a_{ij} x_i x_j \,. \quad (3.4)$$
3. Квадратична — складається з константних, лінійних, добуткових та квадратичних елементів:
$$Y = a_0 + \sum_{i=1}^{N} a_i x_i + \sum_{i \neq j}^{N} a_{ij} x_i x_j + \sum_{i=1}^{N} x_i^2 \,. \quad (3.5)$$
4. Напів-квадратична — складається з константних, лінійних та квадратичних елементів:
$$Y = a_0 + \sum_{i=1}^{N} a_i x_i + \sum_{i=1}^{N} x_i^2 \,. \quad (3.6)$$

Результати прогнозування процесів територіального управління згаданими вище регресійними моделями подано в табл. 3.1.

Таблиця 3.1

Точність прогнозування засобами МЛР

	Лінійна	Добуткова	Квадратична	Напів-квадратична
Коефіцієнт детермінації	0,608	0,695	**0,713**	0,680

3.3.2. МГВА

МГВА [70] застосовується в різноманітних сферах для аналізу даних та знаходження знань та закономірностей, прогнозування і моделювання систем, оптимізації та розпізнавання образів. Індуктивні алгоритми МГВА дають можливість автоматично знаходити взаємозалежності даних, вибрати оптимальну структуру моделі, підвищувати точність наявних алгоритмів. МГВА складається з ряду алгоритмів для вирішення широкого спектра завдань. В нього входять як параметричні алгоритми, так і алгоритми кластеризації, комплексування аналогів, ребінаризації та ймовірнісні алгоритми. Цей підхід самоорганізації базується на переборі моделей, що поступово ускладнюються, та на виборі найкращого розв'язку згідно з мінімумом встановленого критерію. Вхідні дані розділяються на навчальну та тестову множини. Навчальна множина використовується для визначення коефіцієнтів моделі, в той час як тестова множина використовується для визначення точності моделі на базі оцінених коефіцієнтів.

Використано такі види МГВА:

- Комбінаторний МГВА.
- МГВА з активними нейронами.

Комбінаторний МГВА – це поліноміальна функція з лінійними параметрами. Комбінаторний метод складається з добутків поліноміальної функції, яка генерується з множини змінних.

МГВА з активними нейронами, також відомий як поліноміальні нейронні мережі, використовує алгоритм комбінаторної оптимізації нейронних зв'язків. Алгоритм ітеративно створює шари нейронів з двома або більше входами. Алгоритм зберігає тільки обмежений набір оптимально-комплексних нейронів, які називають шириною початкового шару. Кожен новий шар створюється з використанням двох або більше нейронів, отриманих з будь-яких попередніх шарів. Для кожного нейрона у мережі застосовується функція передачі (як

правило, з двох змінних), що дозволяє скінченним комбінаторним пошуком вибрати передавальну функцію, яка найточніше прогнозує дані тестування. Передавальна функція зазвичай має квадратичну або лінійну форму. МГВА з активними нейронами генерує багато шарів, але зв'язки шарів настільки рідкі, що їх число дорівнює числу з'єднань з одного шару.

Експериментальним шляхом було обрано наступні параметри МГВА для досягнення найкращого прогнозування:

- співвідношення навчальної та тестової множини – 50% та 50%;
- зовнішній критерій оцінки – середньоквадратична помилка.

Результати екстраполяції подано в табл. 3.2.

Таблиця 3.2

Точність прогнозування засобами регресії та МГВА

	Лінійна	Добуткова	Квадратична	Напівквадратична	Комбінаторний МГВА	Нейронний МГВА
Коефіцієнт детермінації	0.608	0.695	0.713	0.680	0.700	**0.89**

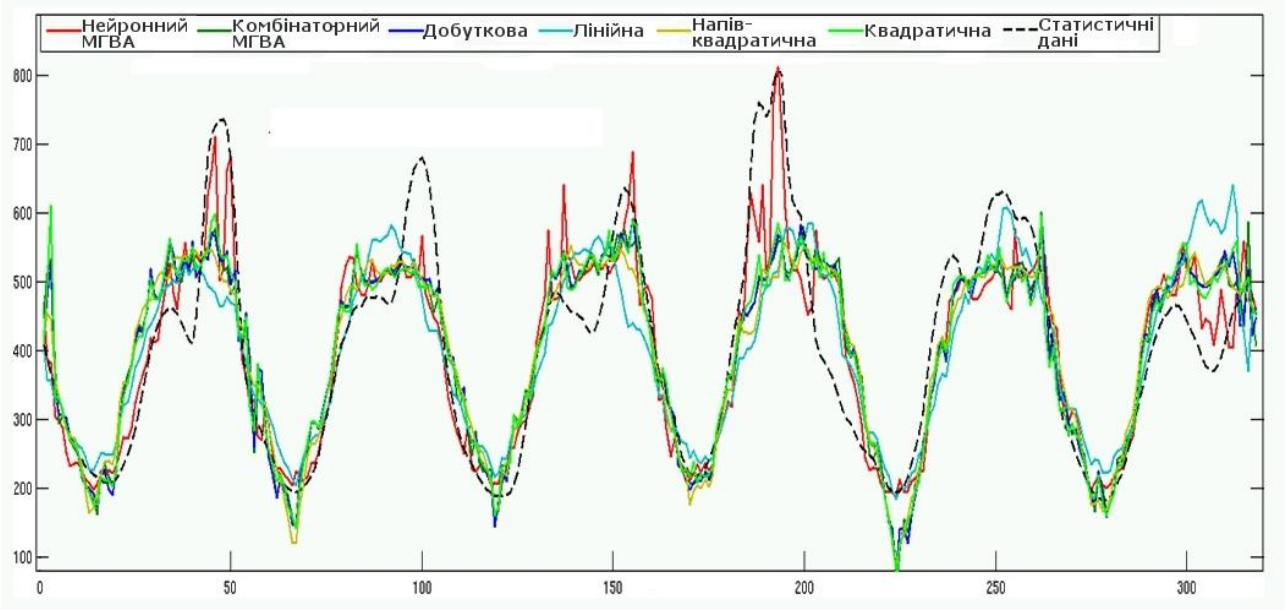


Рис. 3.6. Точність прогнозування засобами регресії та МГВА

3.4. Розроблення засобів інтеграції даних та їх аналізу

Для аналізу просторових даних перш за все необхідно їх інтегрувати. З цією метою використано засоби Microsoft SQL Server, а саме формування гіперкуба даних з використанням Business Intelligence та SQL Server Interated Services та Microsoft Time Series (рис. 3.7) [65].

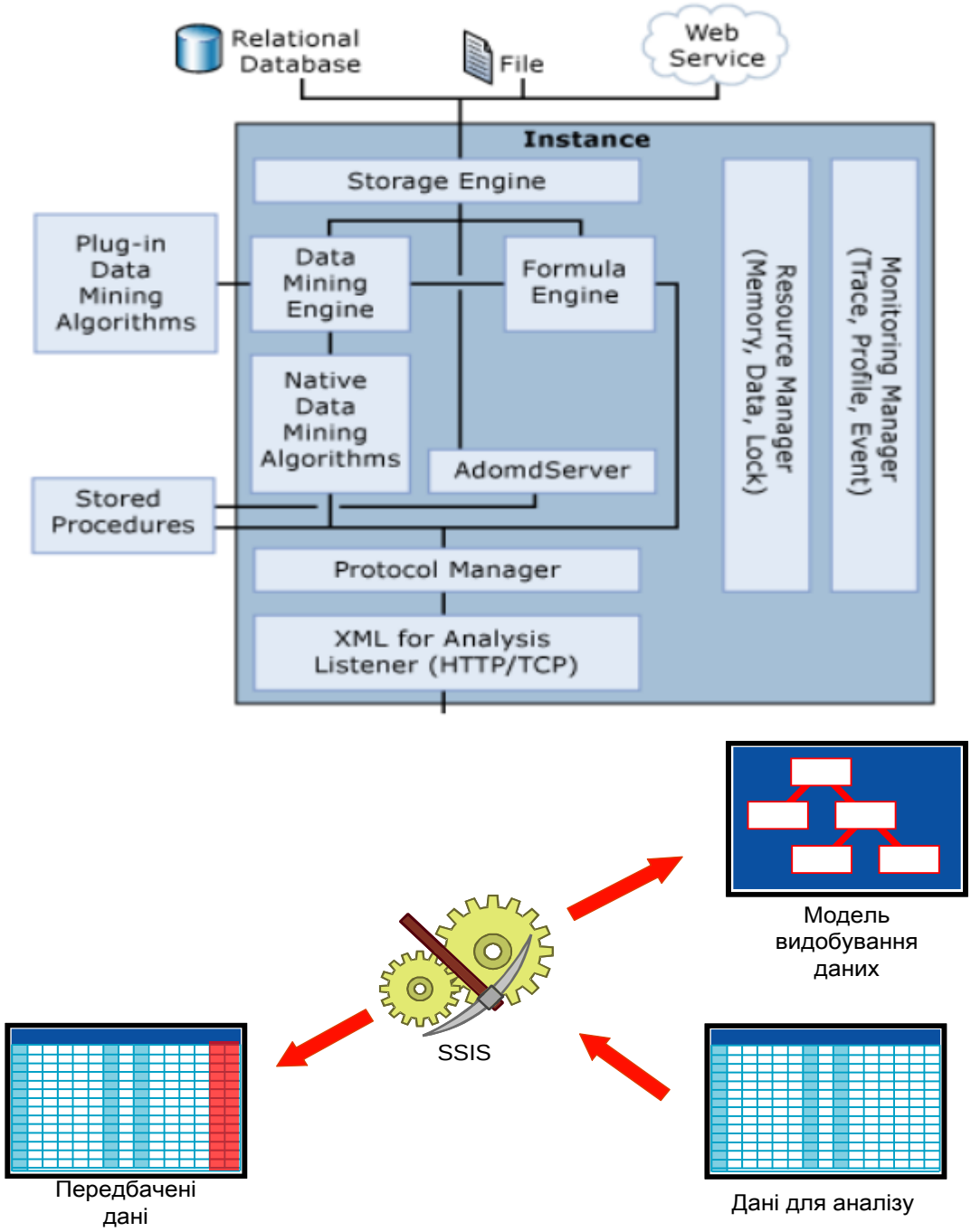


Рис. 3.7. Структура засобів інтеграції та аналізу даних Microsoft SQL Server

Одна з найбільш значних відмінностей SQL Server 2012 порівняно з SQL Server 2008, що допомогає підприємствам впоратися з великими обсягами даних, є його сумісність з Hadoop. Hadoop дозволяє користувачам обробляти великі обсяги структурованих і неструктурованих даних, дає змогу швидко порівняти структури. Оскільки Hadoop є продуктом з відкритим вихідним кодом, він може забезпечити ці знання з нижчою вартістю рішення. Партнерство Microsoft з Hortonworks у розробці сумісності Hadoop з SQL Server 2012, а також нещодавно оголошений спільний сервер Microsoft HDInsight сервер і Windows Azure HDInsight послуг дозволяє клієнтам використовувати Microsoft розвинені роз'єми Hadoop, щоб отримати кращі ідеї із даних. З використанням Driver Hive ODBC, який з'єднує SQL Server з Hadoop, клієнти тепер можуть використовувати такі Microsoft BI інструменти, як PowerPivot і Power View в SQL Server 2012, щоб проаналізувати всі типи даних, у тому числі неструктуровані дані. Крім того, з новими Service Data Quality в SQL Server 2012, клієнти можуть підвищити якість своїх даних шляхом перетворення вихідних даних в достовірні та послідовні серії даних для моделювання.

Алгоритм Microsoft Time Series забезпечує алгоритми регресії, які оптимізовані для прогнозування безперервних значень, таких як продаж продукції протягом тривалого часу. У той час як інші алгоритми Microsoft, такі як дерева рішень, вимагають формування додаткових атрибутів з новою інформацією в якості вхідних даних для прогнозування тренда, модель часових рядів працює лише на вхідних даних. За допомогою моделі часового ряду можна передбачити тенденції, що базуються тільки на оригінальному наборі даних, який використовується для створення моделі. Також передбачена можливість додавати нові дані в модель, коли прогноз вже зроблено, і автоматично включати нові дані в аналіз тренда.

Важливою особливістю алгоритму Microsoft Time Series є те, що він може виконувати крос-прогноз. Якщо здійснюється тренування алгоритму з двома окремими, але пов'язаними серіями, то можна використовувати отриману

модель, щоб передбачити результат однієї серії, заснований на поведінці інших серій. Наприклад, спостережуваний продаж одного продукту може впливати на прогнозований продаж іншого продукту.

Крос-прогноз також корисний для створення загальної моделі, яку можна застосувати до кратних рядів. Наприклад, прогнози для конкретного регіону нестійкі, бо бракує серії якісних даних [120]. Тоді можна навчати загальну модель в середньому на чотирьох регіонах, а потім застосувати модель до індивідуальних серій, щоб створити стабільніші прогнози для кожного регіону.

У SQL Server 2005 до Microsoft Time Series використовувався один алгоритм, ARTXP. Алгоритм ARTXP був оптимізований для короткострокових прогнозів, і тому використовувався для передбачення наступного ймовірного значення в серії. Починаючи з SQL Server 2008, Microsoft Time Series використовує як алгоритм ARTXP, так і другий алгоритм, ARIMA. Алгоритм ARIMA оптимізований для довгострокового прогнозування.

За замовчуванням Microsoft Time Series використовує поєднання обох алгоритмів, коли він аналізує зразки і робить прогнози. Алгоритм навчає дві окремі моделі на тих же даних: одна модель використовує алгоритм ARTXP, інша – алгоритм ARIMA. Потім алгоритм поєднує результати двох моделей для отримання кращого прогнозу за змінним числом часових зрізів. Оскільки ARTXP кращий для короткострокових прогнозів, то він має більшу вагу на початку ряду прогнозування. На дальших етапах більшу вагу отримує ARIMA.

Керування моделями здійснюється за декількома параметрами.

Перший параметр, PREDICTION_SMOOTHING, відповідає за періодичність і використовується для пошуку сезонних залежностей. Наступний необхідний параметр, Prediction Steps, використовується для встановлення віддалі прогнозу.

Висновки до розділу 3

1. Розроблено метод перетворення реляційних даних у модель «сутність-характеристика», що дає змогу здійснювати федеративне опрацювання даних.
2. Спроектовано схему сховища даних та обрано метод прогнозування значень показників розвитку регіону. Прогнозування розвитку регіону здійснюється методом часових рядів на основі аналізу попередніх станів регіону. Перевірку однорідності даних здійснено на основі критерію Ірвіна.
3. Розроблено універсальну структуру інтелектуального аналізу мультиструктурної інформації, яка дає змогу аналізувати федеративні дані без використання словника даних.

Матеріали розділу опубліковано у [6, 10, 15].

РОЗДІЛ 4. РОЗРОБЛЕННЯ АРХІТЕКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА АПРОБАЦІЯ РЕЗУЛЬТАТІВ

У розділі спроектовано схему даних регіону. Апробовано розроблені методи і алгоритми.

4.1. Розроблення архітектури системи прогнозування розвитку регіону

Архітектура Великих даних з врахуванням розроблених вище методів перетворення даних в модель «сутність-характеристика» та відповіді на запит користувача подана на рис. 4.1.

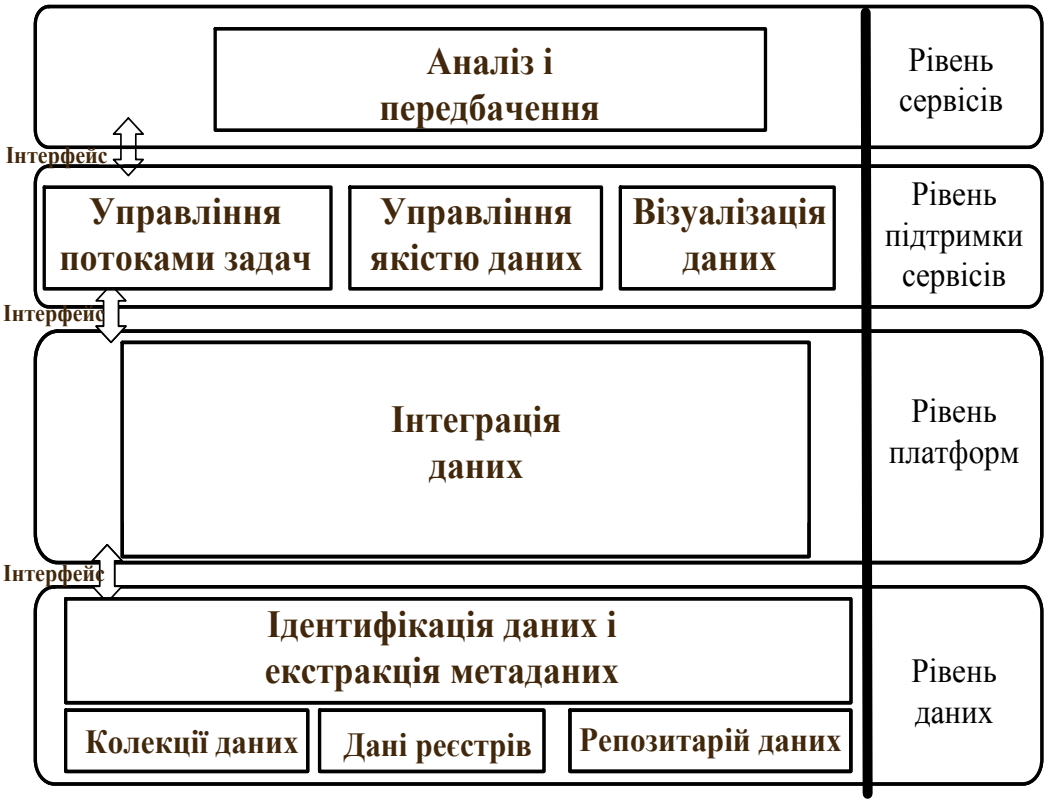


Рис. 4.1. Архітектура системи аналізу даних розвитку регіону

Далі спроектуємо структуру сховища даних для збереження показників розвитку регіону і його елементів (рис. 4.2). Ця структура дає змогу зберігати як деталізовану, так і агреговану інформацію про основні показники регіону.

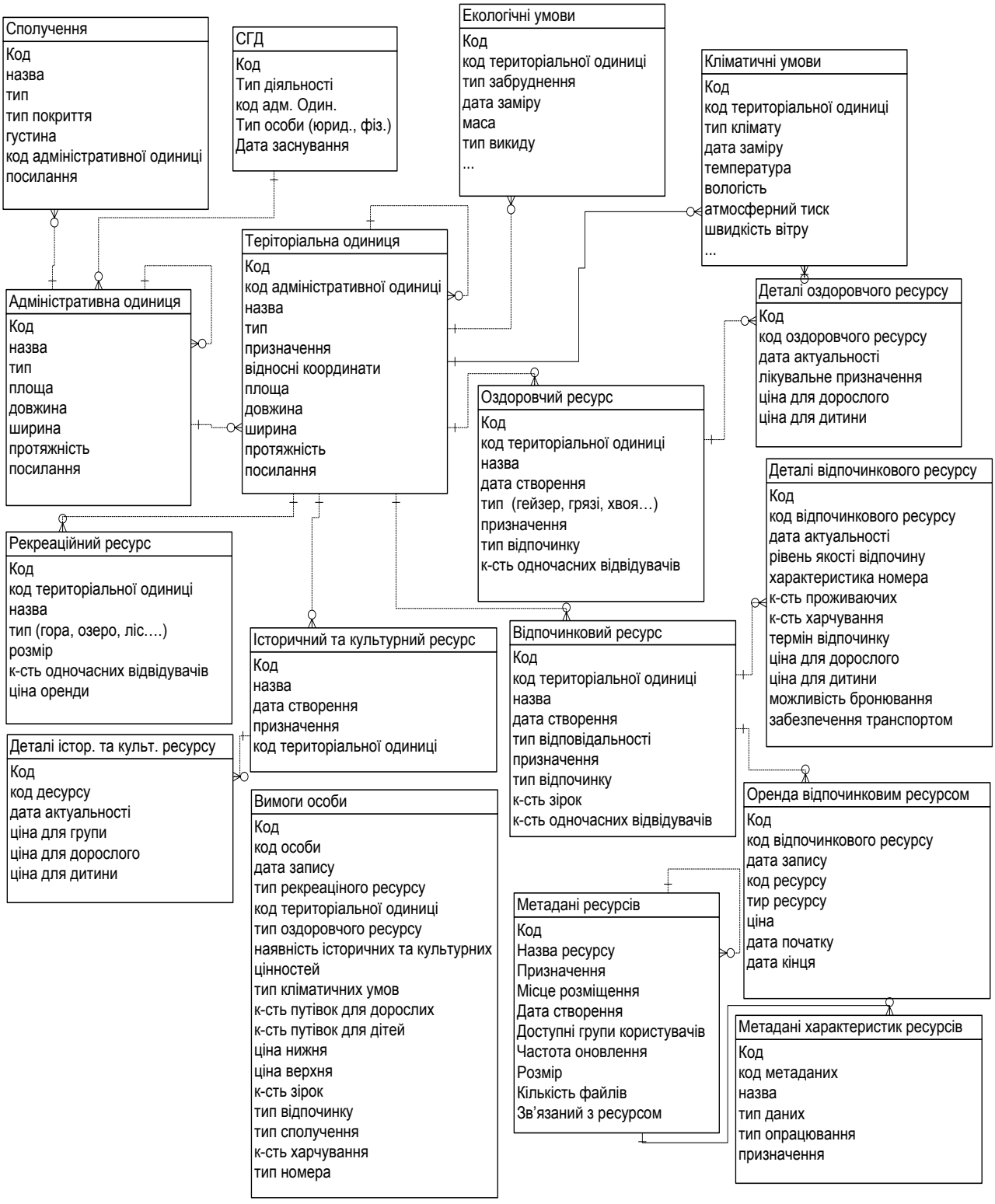


Рис. 4.2. Логічна модель даних регіону

Діаграма класів складається з множини елементів, які в сукупності відображають декларативні знання про предметну область. Ці знання інтерпретуються в базових поняттях мови UML, таких як класи, інтерфейси і відношення між ними та їх складовими компонентами. При цьому окремі компоненти цієї діаграми можуть утворювати пакети для відображення загальнішої моделі системи. Якщо діаграма класів є частиною деякого пакету, то її компоненти мають відповідати елементам цього пакету, включаючи можливі посилання на елементи з інших пакетів.

Діаграма станів описує процес зміни станів тільки одного класу, а точніше – одного екземпляра певного класу, тобто моделює всі можливі зміни у стані конкретного об’єкту. При цьому зміна стану об’єкту може бути викликана зовнішніми діями з боку інших об’єктів або ззовні. Саме для опису реакції об’єкту на подібні зовнішні дії і використовуються діаграми станів.

Головне призначення цієї діаграми – описати можливі послідовності станів і переходів, які в сукупності характеризують поведінку елемента моделі протягом його життєвого циклу. Діаграма станів подає динамічну поведінку сутності на основі специфікації її реакції на сприйняття деяких конкретних подій. Системи, які реагують на зовнішні дії від інших систем або від користувачів, іноді називають реактивними. Якщо такі дії ініціюються в довільні випадкові моменти часу, то говорять про асинхронну поведінку моделі [12].

Діаграма класів описує представлення даних в моделі «сутність-характеристика» (рис. 4.3).

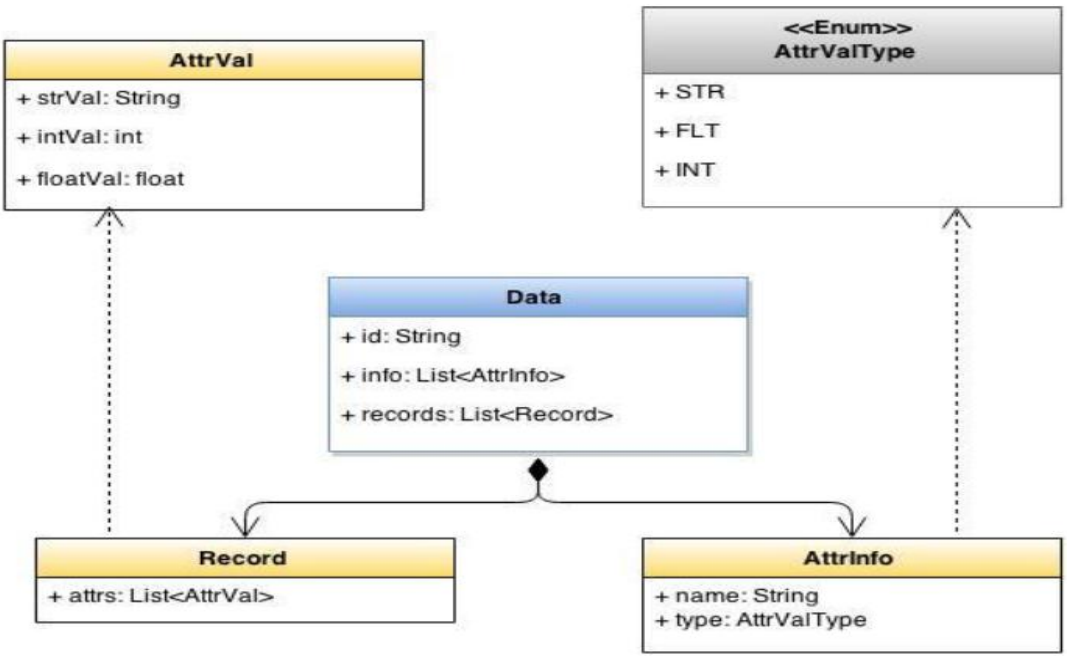


Рис. 4.3. Діаграма класів

Для уніфікованого перетворення запитів розроблено діаграму класів (рис. 4.4) та інтерфейс взаємодії (рис. 4.5).

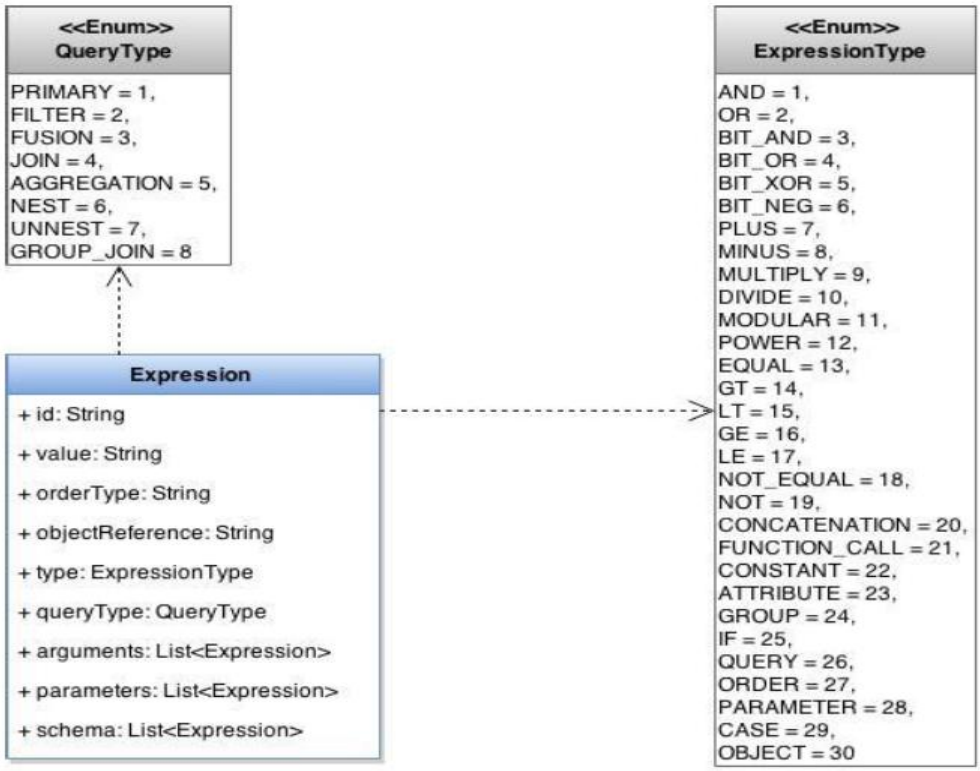


Рис. 4.4. Діаграма для уніфікованого перетворення запитів

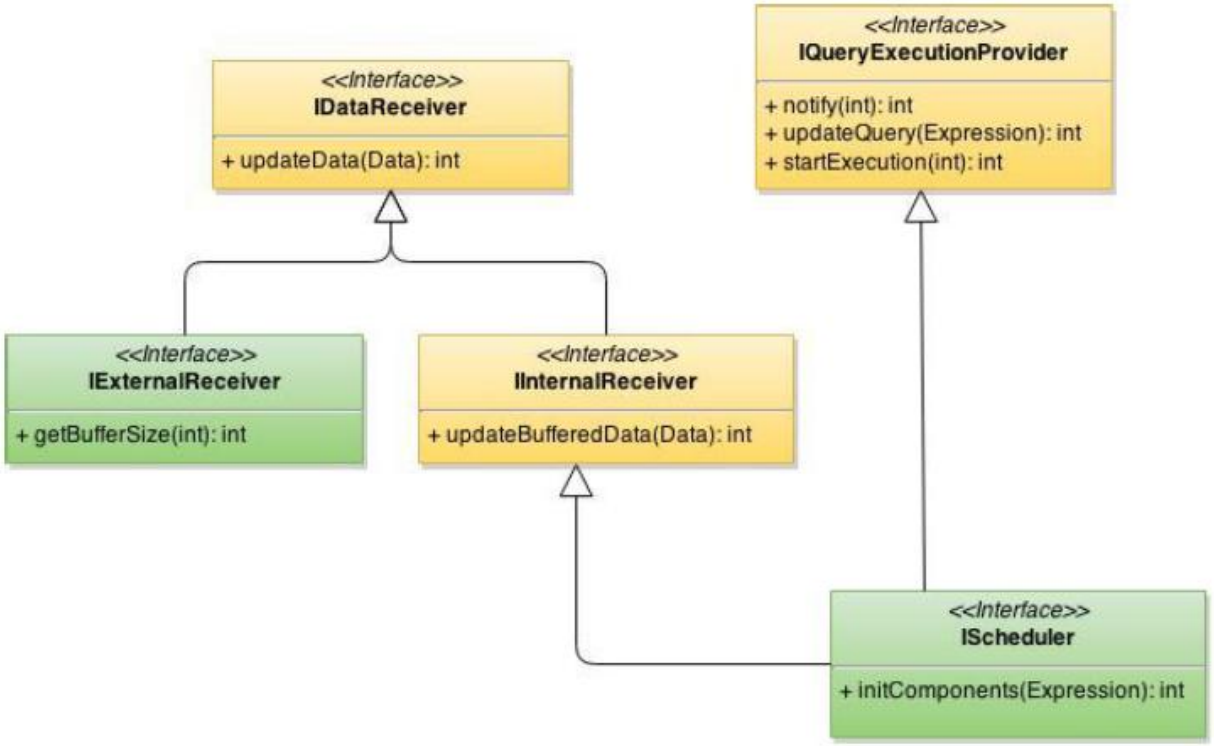


Рис. 4.5. Інтерфейс для уніфікованого перетворення запитів

Діаграма діяльності. Для моделювання процесу виконання операцій в мові UML використовуються так звані діаграми діяльності. Вживана в них графічна нотація багато в чому схожа на нотацію діаграми станів, оскільки на діаграмах діяльності також присутні позначення станів і переходів. Відмінність полягає в семантиці станів, які використовують для подання не діяльностей, а дій, і у відсутності на переходах сигнатури подій. Кожне перебування на діаграмі діяльності відповідає виконанню деякої елементарної операції, а перехід в наступний стан виконується тільки після завершення операції в попередньому стані. Діаграма діяльності подається у формі графа діяльності, вершинами якого є стани дій, а дугами – переходи від одного стану дії до іншого.

Основним напрямом використання діаграм діяльності є візуалізація особливостей реалізації операцій класів, коли необхідно зобразити алгоритми їх виконання. При цьому кожний стан може бути виконанням операції деякого

класу або її частини, дозволяючи використовувати діаграми діяльності для опису реакцій на внутрішні події системи [12].

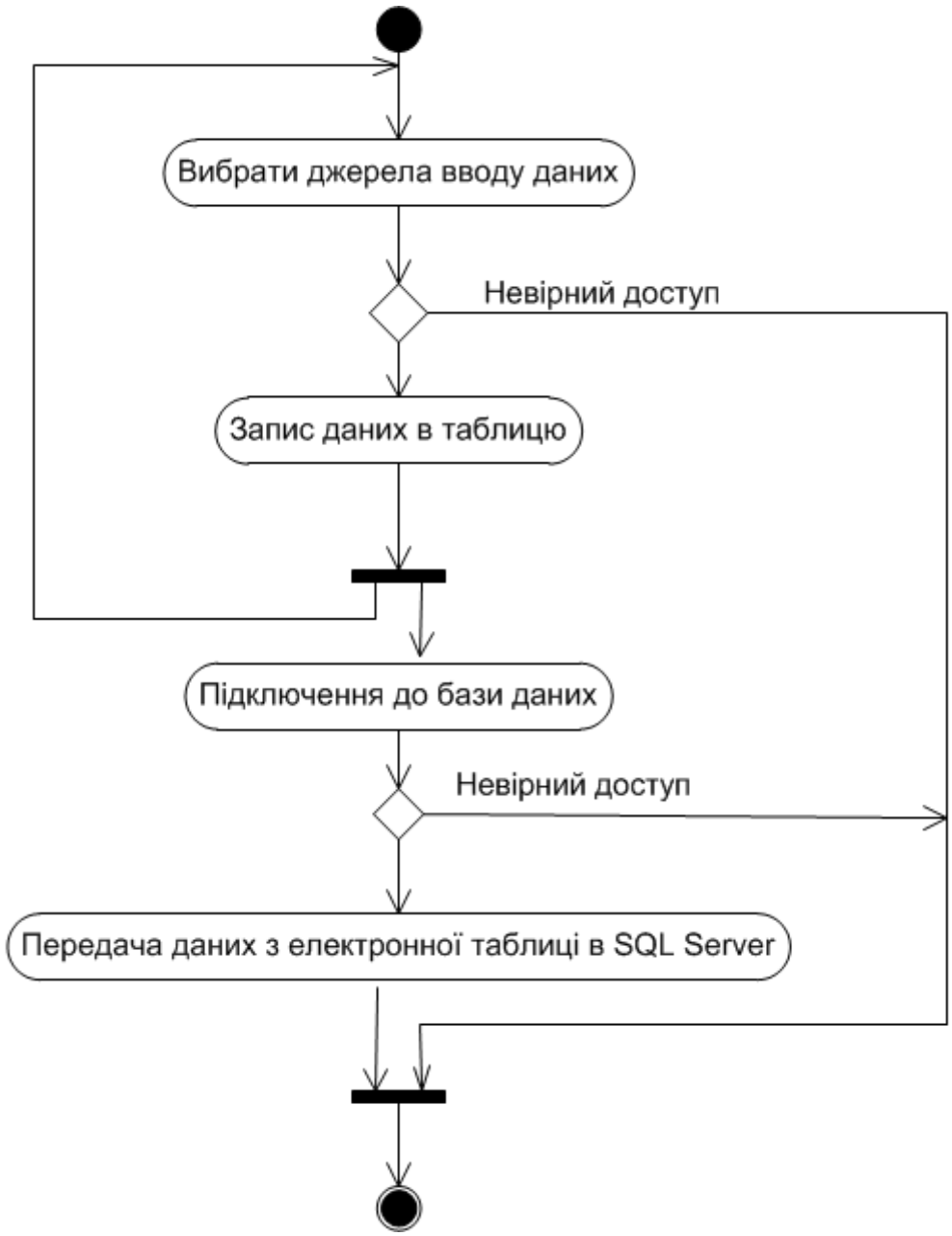


Рис. 4.6. Діаграма діяльності

Діаграма послідовності. На діаграмі послідовності зображуються винятково ті об’єкти, які безпосередньо беруть участь у взаємодії і не показуються можливі статичні асоціації з іншими об’єктами. Для діаграми послідовності ключовим моментом є саме динаміка взаємодії об’єктів у часі. При цьому діаграма послідовності має два виміри. Один – зліва направо у

вигляді вертикальних ліній, кожна з яких зображає лінію життя окремого об’єкта, що бере участь у взаємодії. Графічно кожний об’єкт зображується прямокутником і розташовується у верхній частині своєї лінії життя. Всередині прямокутника записуються ім’я об’єкта й ім’я класу, розділені двокрапкою. При цьому весь запис підкреслюється, що є ознакою об’єкту, який, як відомо, є екземпляром класу.

Крайнім ліворуч на діаграмі зображується об’єкт, що є ініціатором взаємодії. Правіше зображується інший об’єкт, що безпосередньо взаємодіє з першим. Таким чином всі об’єкти на діаграмі послідовності утворюють деякий порядок, обумовлений ступенем активності цих об’єктів під час взаємодії один з одним.

Другий вимір діаграми послідовності – вертикальна часова вісь, спрямована зверху вниз. Початковому моменту часу відповідає найвища частина діаграми. При цьому взаємодію об’єктів реалізують за допомогою повідомлень, які посилають одні об’єкти іншим. Повідомлення зображують у вигляді горизонтальних стрілок з іменем повідомлення, які також утворюють порядок відповідно до часу свого виникнення. Інакше кажучи, повідомлення, розташовані на діаграмі послідовності вище, ініціюються раніше від тих, які розташовані нижче. При цьому масштаб на осі часу не вказується, оскільки діаграма послідовності моделює лише тимчасову впорядкованість взаємодій типу «пізніше» [12].

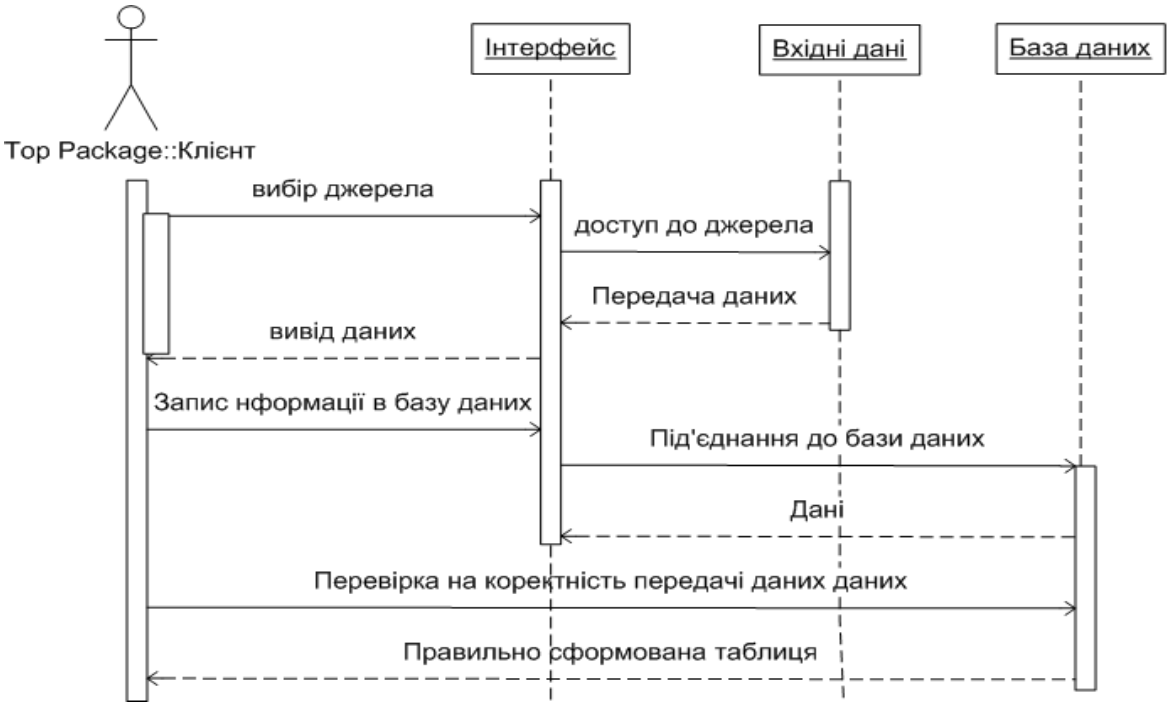


Рис. 4.7. Діаграма послідовності

Діаграма компонентів. Діаграма компонентів, на відміну від раніше розглянутих діаграм, описує особливості фізичного подання системи. Діаграма компонентів дозволяє визначити архітектуру системи, що розробляється, встановивши залежності між програмними компонентами, в ролі яких може виступати початковий, бінарний і виконуваний код. У багатьох середовищах розроблення модуль або компонент відповідає файлу. Пунктирні стрілки, що з'єднують модулі, показують відношення взаємозалежності, аналогічні до тих, які мають місце при компіляції початкових текстів програм. Основними графічними елементами діаграми компонентів є компоненти, інтерфейси і залежності між ними.

Діаграма компонентів розробляється для таких цілей:

1) візуалізації загальної структури початкового коду програмної системи;
2) специфікації здійсненого варіанту програмної системи;
3) забезпечення багатократного використання окремих фрагментів програмних кодів;

4) подання концептуальної і фізичної схем баз даних.

У розробленні діаграм компонентів беруть участь як системні аналітики й архітектори, так і програмісти. Діаграма компонентів забезпечує узгоджений перехід від логічного подання до конкретної реалізації проекту у формі програмних кодів. Одні компоненти можуть існувати тільки на етапі компіляції програмних кодів, інші – на етапі його виконання. Діаграма компонентів відображає загальні залежності між компонентами, розглядаючи останні як класифікатори.

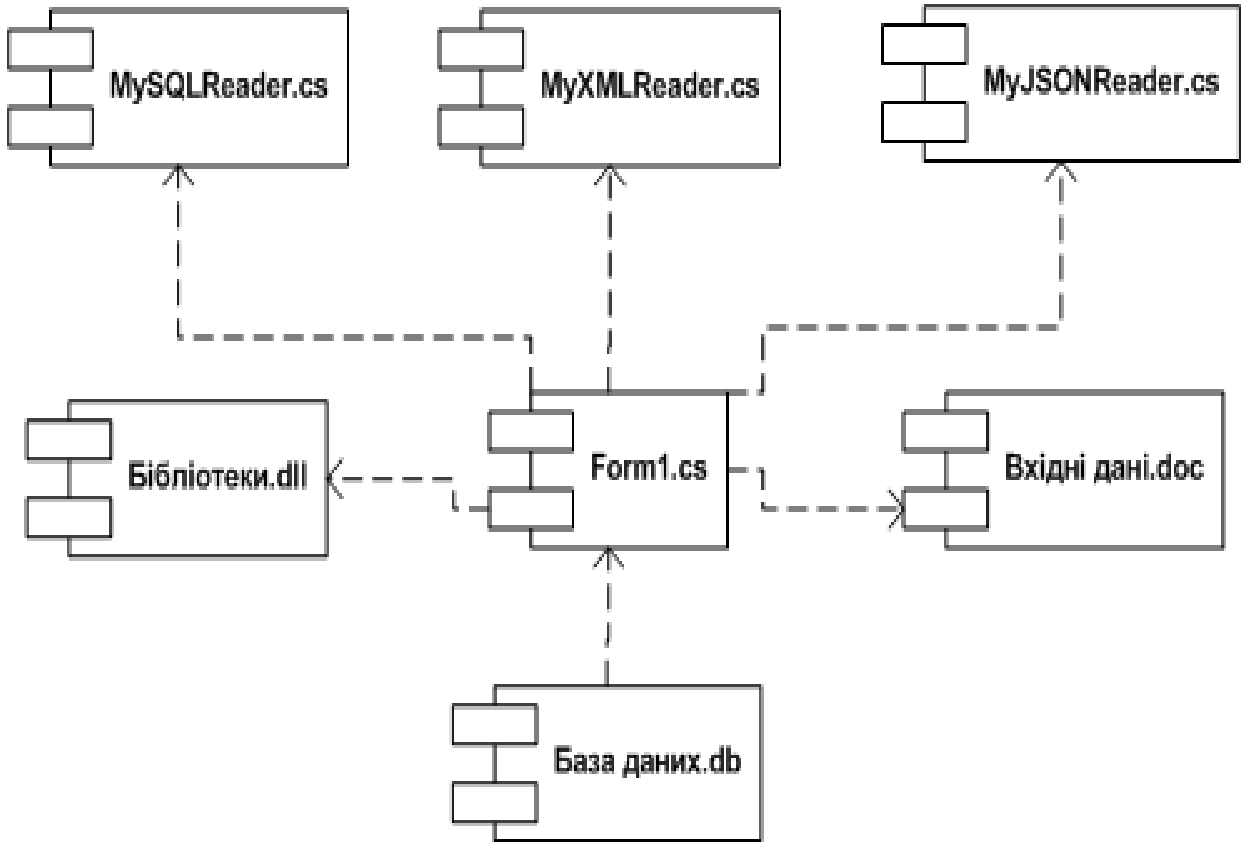


Рис. 4.8. Діаграма компонентів

Аналіз регіону у програмі проводиться по окремих галузях. Домінуючими галузями вибрані економіка, промисловість, екологія та соціологія. Це представлено на рис. 4.9. Для кожної галузі існує окремий набір більш детермінованих критеріїв, за якими виводиться комплексна оцінка.

Ці критерії визначаються розробником ПЗ на основі наявних відомостей про кожну досліджену галузь. Вони не є рівноправними. Кожен критерій повинен мати власну важливість для сумарної оцінки. У даному ПЗ усі критерії мають коефіцієнти важливості, значення яких лежить у межах від 0 до 1.

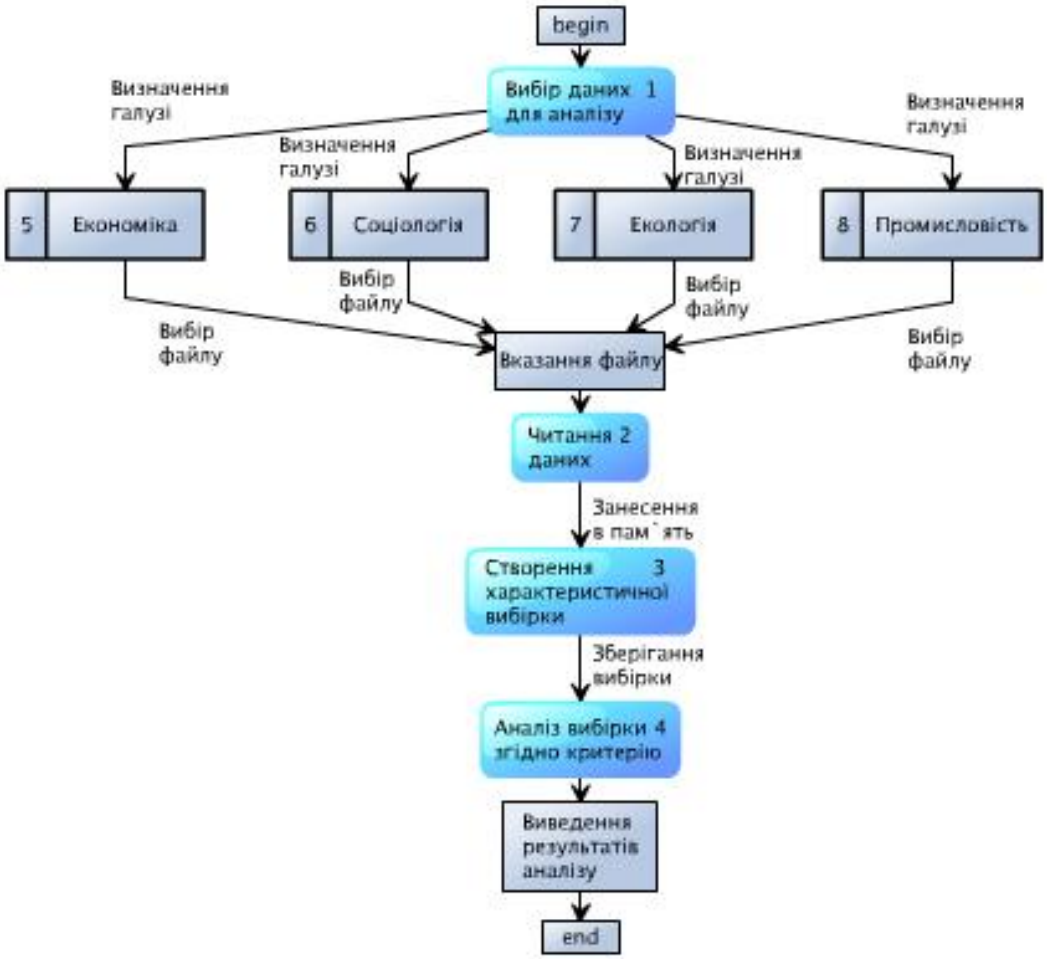


Рис. 4.9. Загальна схема роботи розробленого ПЗ

Це продемонстровано на прикладі першої галузі – економіки. Коефіцієнти мають власну вагу та їх сума наближено дорівнює 1. Згідно цих коефіцієнтів (рис. 4.10), визначені статистично значення кожного критерію приводяться до кінцевих результатів. Сума отриманих оцінок визначає розвиток цілої галузі у регіоні.

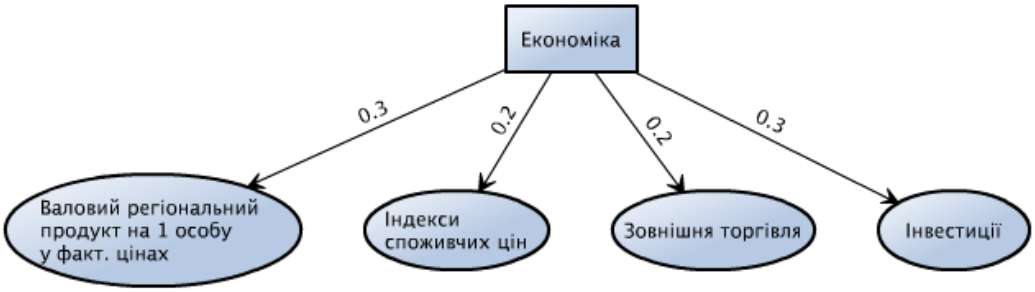


Рис . 4.10. Критерії оцінювання галузі «Економіка»

Конкретно для економіки головними критеріями були визначені валовий регіональний продукт, індекси споживчих цін, показники зовнішньої торгівлі та інвестиції. Кожен з них характеризує частину економічної ситуації. І в кожному з них нас цікавить не стільки показник, скільки його динаміка. Динаміку можна визначити, дізнавшись різниці між показниками за певний період часу і взявши їх середнє значення.

На основі показників динаміки із звичайного статистичного аналізу можна перейти до прогнозування. Згідно показникам побудуємо тренд розвитку галузі.

Щодо промисловості, можемо визначити такі ключові фактори, як об`єм реалізованої продукції, розміри роздрібної торгівлі та об’єми вантажних перевезень. Ця структура показана на схемі рис. 4.11 і на ній визначені коефіцієнти. Обгрунтуванням рівності коефіцієнтів виступає незамінність кожного фактора у загальній схемі організації промисловості.

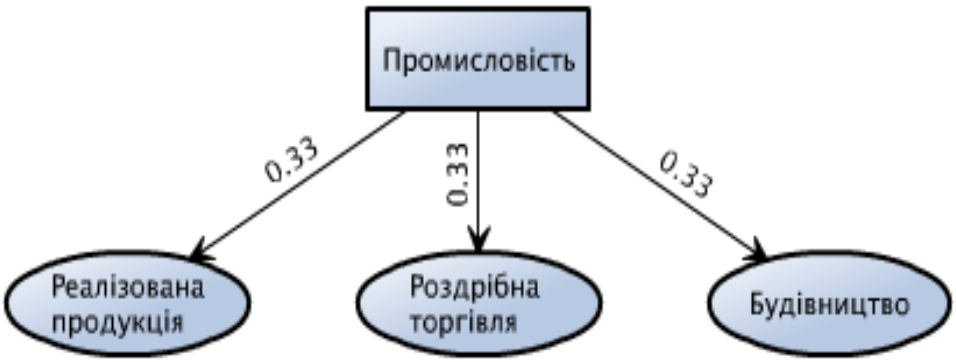


Рис. 4.11. Критерії оцінювання галузі «Промисловість»

Дані по цих критеріях можна отримати з офіційних та урядових джерел, на сайтах державних служб і т.д.

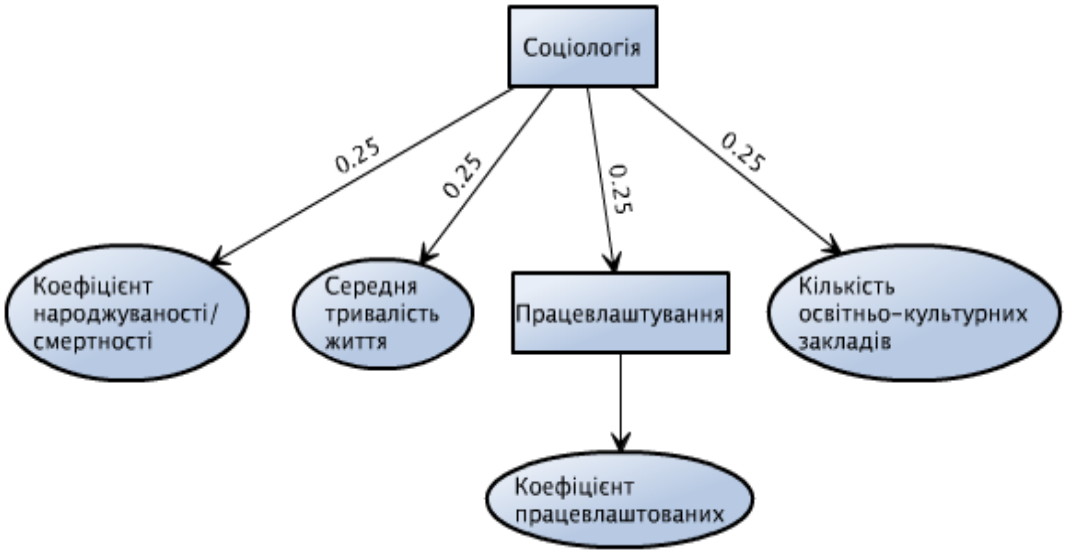


Рис. 4.12. Критерії оцінювання галузі «Соціологія»

Соціологічна складова є головним критерієм рівня життя населення. Такі параметри як коефіцієнт народжуваності/смертності характеризує природній приріст населення і безпосередньо впливає на кількість населення регіону. Середня тривалість життя визначається рядом факторів з соціологічної, економічної, екологічної сфер та може служити індикатором рівня життя, рівень працевлаштування пов’язаний з економічною сферою вкрай тісно і характеризує робочі місця, що є у наявності на підприємствах.

Індикатором рівня розвитку культури та науки є кількість навчально-виховних закладів або кількість спеціалістів, що випускаються з ВНЗ кожного року. За оцінку виберемо перший показник, оскільки другий може бути неточним або необґрунтованим.

Соціологія охоплює значну частину подій та сфер життя регіону. Від святкувань у масштабах регіону до забезпечення житлових будинків комунальними послугами – все відображається соціологічними даними. У

такому різноманітті важко визначити домінуючі критерії. Тому призначаємо усім критеріям рівний пріоритет. Визначимо детерміновані показники та методом середнього визначимо приріст кожного показника і, таким чином, отримаємо динаміку зміни усіх критеріїв, за якими і сформуємо прогноз.

Останній набір критеріїв відведемо під екологічні чинники. Більшість людей не приділяють цьому достатньої уваги. Екологія має великий вплив на соціологію, але менший ніж на промисловість.

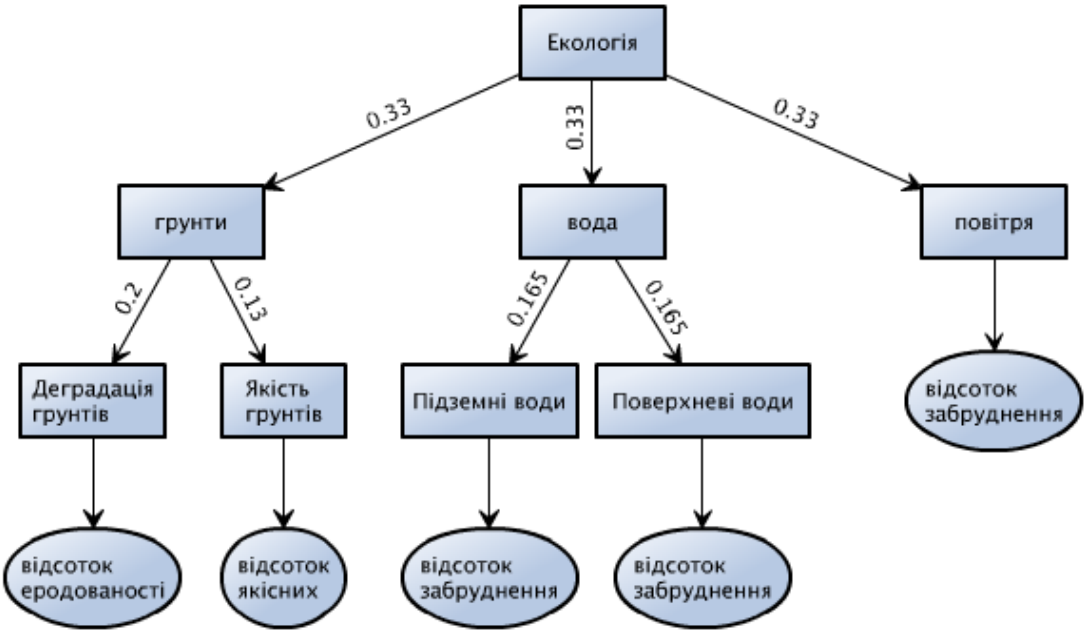


Рис. 4.13. Критерії оцінювання галузі «Екологія»

Екологія впливає на стан здоров'я і середню тривалість життя, на стан розвитку сільського господарства, у важких екологічних умовах стає невигідним виробництво. Загальний екологічний стан можна характеризувати трьома такими компонентами: стан грунтів, стан водних ресурсів та стан повітря.

Стан грунтів характеризується еродованістю (забрудненням) та початковою якістю (плодючістю) грунту. При цьому, неправильна обробка грунтів може знищити існуючий баланс, тому критерій відсотків забруднення отримає більший коефіцієнт важливості ніж якість грунту.

Щодо водних ресурсів можна сказати, що вони мають бути логічно розділені на 2 типи: поверхневі і підземні. Поверхневі води – це водойми, що знаходяться на поверхні і безпосередньо залежать від стану повітря та опадів (що попадають туди без фільтрації). Підземні води також надзвичайно залежать від опадів, але вода до них надходить частково відфільтрована шарами грунту. Кожен з цих типів водойм має окремий показник забрудненості, який ми і візьмемо за характеристику.

Останнім параметром виступає чистота повітря. Забрудненість повітря здійснюється технікою і заводами, але результатом стає накопичення шкідливих газів у атмосфері. Через однаковість результату (різниця полягає лише у кількісному значенні) ми поєднаємо усі «забрудники» і використаємо загальний параметр «забрудненість повітря». Кожна з характеристик має вагомий вплив на життя у регіоні, на загальний рівень комфорту населення, тому присвоїмо їм однакові коефіцієнти важливості.

Використовуючи статистику як науку, не можна оминути важливий фрагмент розрахунку параметрів – визначення достовірності розрахунків. Необхідно для кожного показника, вираженого у відсотках, вирахувати похибку за формулою

$$m = \sqrt{\frac{p \cdot q}{n}} \text{ або } m = \sqrt{\frac{p \cdot q}{n-1}} \text{ для } n<30, p=\text{показник}, q=(100-p). \quad (4.1)$$

Це дозволить говорити про рівень достовірності кожної окремої оцінки. Загальний рівень достовірності визначимо середнім рівнем похибки усіх оцінок.

В результаті за усіма окресленими даними створимо програмне забезпечення, що аналізує ряд конкретних параметрів основних процесів розвитку регіону і наводить тенденції розвитку кожного з них. За отриманими результатами можна скласти прогноз розвитку регіону з урахуванням його попередньої динаміки.

4.2. Програмна реалізація та аналіз результатів прогнозування розвитку регіону

Для визначення методики аналізу фінансових показників регіону розроблене відповідне програмне забезпечення.

Для реалізації сховища даних була вибрана 64-розрядна платформа (x64). Узагальнене сховище даних складається з наступних компонент:

- Реляційна база даних (Microsoft SQL Server Database Services, Oracle Database, MySQL,PostgreSQL) у 64-розрядному виконанні;
- Багатовимірна база даних (Microsoft SQL Server Analysis Services або Hyperion Essbase) у 64-розрядному виконанні;
- «Ієрархічна» база даних (MongoDB);
- Система керування федеративним сховищем даних, яка є окремою програмою, розробленою спеціально для забезпечення функціонування сховища даних, та включає в себе файлове сховище.

Для розроблення системи керування федеративним сховищем даних було використано платформу Microsoft.Net, мову C# та середовище розробки Visual Studio. Бібліотека класів, що постачається з .Net та мова високого рівня C# , а також методологія RAD (швидкої розробки застосувань), на якій побудоване середовище розробки Visual Studio дозволяє швидко створювати застосування, орієнтовані на бази даних.

В першу чергу здійснюється федеративне опрацювання даних з джерел. Аналізуємо відносну кількість об’єктів або документів, наявних у джерелах даних, до загальної кількості об’єктів, які потрапили у федеративне сховище. У таблиці нижче наведена структура детермінованих схем БД з деталізацією за областями в обсязі, достатньому для прогнозування процесів розвитку регіону.

В ній також вказана структурованість даних і розміщення у сховищі даних (порядковий номер області та тип БД).

Таблиця 4.1

Структура детермінованих схем баз даних

Джерело/область/таблиця	Відстань $V(e, f)$	Розміщення
Джерело даних MFUVUDB	**0.154224**	-
MFUVUDB.t_aspnet_Roles	0	RDB
MFUVUDB.t_aspnet_Applications	0	RDB
MFUVUDB.t_mfuvu_Educ_types	1	RDB
MFUVUDB.t_mfuvu_Settlements	1	RDB
MFUVUDB.t_vw_mfuvu_list_sti_correspond	0.666942	RDB, XML
MFUVUDB.t_vw_mfuvu_Users	0.401167	RDB, XML
Джерело даних TransBudgDB	**0.628414**	-
TransBudgDB.t_VXXDDMMYY	0.961365	RDB
TransBudgDB.t_D_BUDG_LOCAL_DET	0.888913	RDB
TransBudgDB.t_D_ECON_CRED	0.9	RDB, XML
TransBudgDB.t_D_FIN	0.916667	RDB, XML
TransBudgDB.t_DMYYMM	0.533333	RDB, XML
TransBudgDB.t_D_INC_DET	0.903839	RDB
TransBudgDB.t_district	1	RDB
TransBudgDB.t_obl_region	1	RDB
TransBudgDB.t_INCDET	1	RDB
TransBudgDB.t_DOV_TAX	1	RDB
TransBudgDB.t_VW_EXPENSES_DET	0.5	RDB, XML
TransBudgDB.t_VW_EXPENSES_LVL1	0.5	RDB, XML
TransBudgDB.t_VW_EXPENSES_LVL3	0.5	RDB, XML
TransBudgDB.t_VW_INCOME_DET	0.5	RDB, XML
TransBudgDB.t_VW_INCOME_LVL1	0.5	RDB, XML
TransBudgDB.t_VW_INCOME_LVL3	0.5	RDB, XML
Область даних FUPortalDB	**0.325218**	-
FUPortalDB.t_Users	1	RDB
FUPortalDB.t_Applications	0.666667	RDB, XML
FUPortalDB.t_IsAuthCorrespond	1	RDB

Таблиця 4.1 (продовження)

FUPortalDB.t_Articles	0.375858	RDB, XML
FUPortalDB.t_Sections	1	RDB
FUPortalDB.t_People	0.227109	RDB, XML
FUPortalDB.t_Conclusions	1	RDB
FUPortalDB.t_ArticleStates	1	RDB
FUPortalDB.t_aspnet_Applications	0.666667	RDB, XML

Далі розроблено засоби трансформації запитів у різних моделях даних. Результат трансформації подано на рис. 4.14.

```
select * from Persons where age >= 22 and age <= 23
```


```
"1": {"str": "filterExpression"},
"2": {"i32": 28},
"9": {
    "lst": [
      "rec",
      1,
      {
        "1": {"str": ""},
        "2": {"i32": 15},
        "9": {
          "lst": [
            "rec",
            2,
            {
              "1": {"str": "age"},
              "2": {"i32": 23},
              "5": {"str": "age"}
            },
            {
              "1": {"str": ""},
              "2": {"i32": 22},
              "5": {"str": "30"}
            }
          ]
        }
      }
    ]
  }
```


Рис.4.14. Результат трансформації запитів у різних моделях даних

Проаналізована повнота потрапляння інформації у федералізоване сховище даних (рис. 4.15).

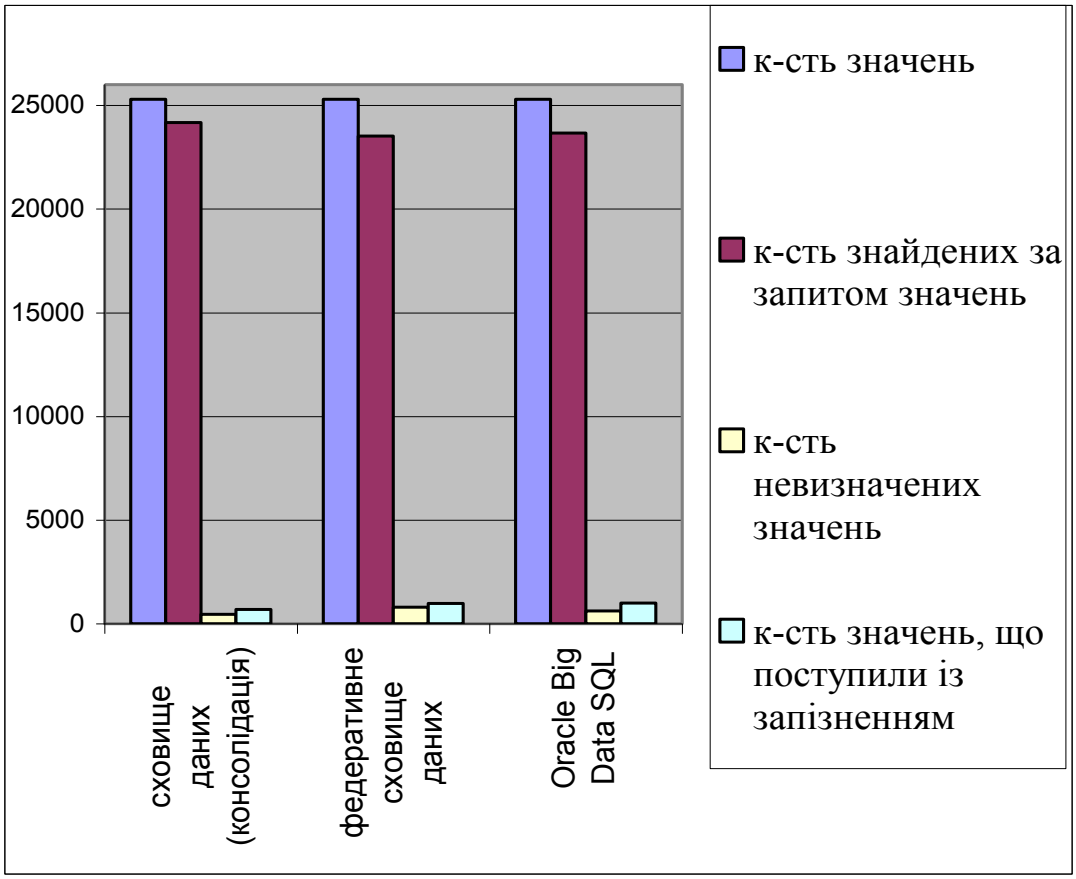


Рис. 4.15. Аналіз повноти накопичених описів об’єктів

На поданій діаграмі проаналізовано роботу алгоритму федеративного запиту. Роботу алгоритму порівняно з роботою алгоритму немодифікованої інтеграції, застосованому в Oracle Data Integrator. Дані у федеративне сховище потрапляють з баз даних різних установ, структури даних яких наперед невідомі. Кількість записів вхідних баз даних, що мають потрапити у федеративне сховище даних – 15000.

Для аналізу повноти накопичених об’єктів порівнювалися традиційне сховище (використано Integrated services), Oracle Big Data SQL та запропоноване рішення. Визначено, що традиційна консолідація працює найкраще, що є зрозумілим, але цей підхід неприйнятний для Великих даних. Тому далі порівнювались запропоноване рішення та Oracle Big Data SQL.

Визначено, що незначно домінується по кількості невизначених значень, але домінує по кількості вірно знайдених даних за запитом, а також дає змогу знайти необхідні дані за коротший час (дані, що поступили з запізненням).

Далі проаналізовано правильність перетворення запитів різних типів.

З цією метою порівнювалися розроблені засоби в системі Інтегратор з середовищем DocumentDB, у якому є можливість формування запитів на мові SQL та NoSQL (рис. 4.16). Результати порівняння подано на рис. 4.16 у відсотковому значенні правильно поданих запитів. Загальна кількість запитів, що тестувались – по 50 запитів кожної категорії. Правильність формування запитів перевірялася експертно.

	Інтегратор				DocumentDB			
вид запиту	select	select…join	insert	delete	select	select…join	insert	delete
РБД (Microsoft SQL Server Database Services)	98	95	93	93	98	92	94	94
XML (XBase)	86	82	-	-	-	-	-	-
NoSQL (mongoDB)	91	86	84	84	89	82	81	81

Рис. 4.16. Результати порівняння інформаційної системи Інтегратор з середовищем DocumentDB

Далі для апробації та тестування розроблених методів визначалося важливість об’єктів у різних джерелах даних. Під важливістю ми розуміли релевантність запиту. Як і слід було очікувати, для реляційних джерел вона вища (рис. 4.17, 4.18).

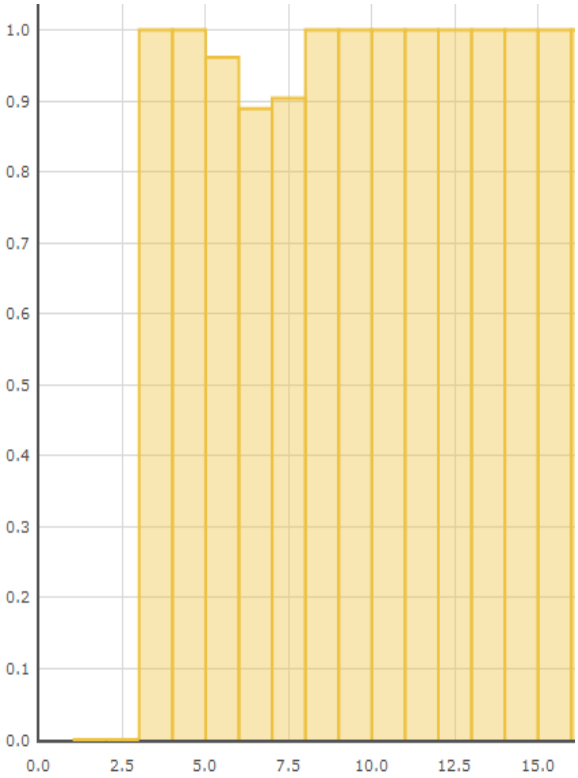


Рис. 4.17. Важливість сутностей в реляційному джерелі

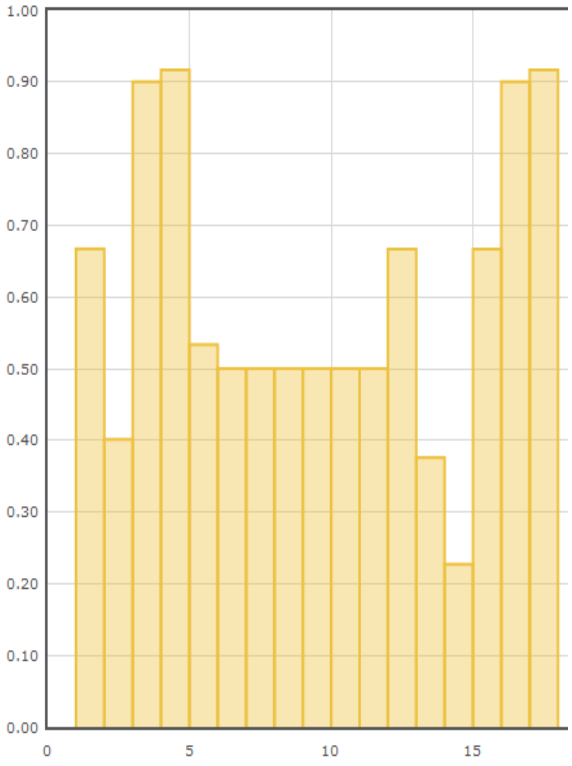


Рис. 4.18. Важливість сутностей в XML джерелі

Наступним кроком є формування інтегрованих показників для прогнозування розвитку регіону. Розроблено засоби для таких показників:

1) коефіцієнт зносу основних засобів,
2) рентабельність активів,
3) фондовіддача основних засобів,
4) коефіцієнт оборотності активів,
5) коефіцієнт фінансування,
6) коефіцієнт платоспроможності,
7) загальний коефіцієнт покриття,
8) коефіцієнт поточної (загальної) ліквідності,
9) коефіцієнт покриття,
10) коефіцієнт абсолютної ліквідності (платоспроможності),
11) частка власних оборотних коштів у покритті запасів,

12) коефіцієнт забезпечення власними коштами,
13) коефіцієнт фінансової автономії,
14) коефіцієнт швидкої ліквідності,
15) коефіцієнт абсолютної ліквідності,
16) чистий оборотний капітал,
17) коефіцієнт платоспроможності (автономії),
18) коефіцієнт забезпеченості власними оборотними коштами,
19) коефіцієнт маневреності власного капіталу.

Для визначення цих показників отримуємо дані з джерел.

Результатом інтеграції даних є формування відношень:

1) Показник – перелік показників з вказуванням їх типу та коефіцієнта важливості.
2) Znach_pokazn – значення коефіцієнтів за роками.
3) Забезпеч_витрат_платежів – значення показників за роками.
4) Діагностика – напрями діагностики (інфраструктура, фінанси та ін.).
5) Рівні – розподіл рівнів діагностики за напрямами.
6) Показник_діагностики – перелік показників діагностики та їх одиниць вимірювання.
7) Знач_показника_діагностики – значення показників діагностики за роками.

У першу чергу побудуємо діаграму зміни показників (кнопка «Діаграма зміни показника») (рис. 4.19):

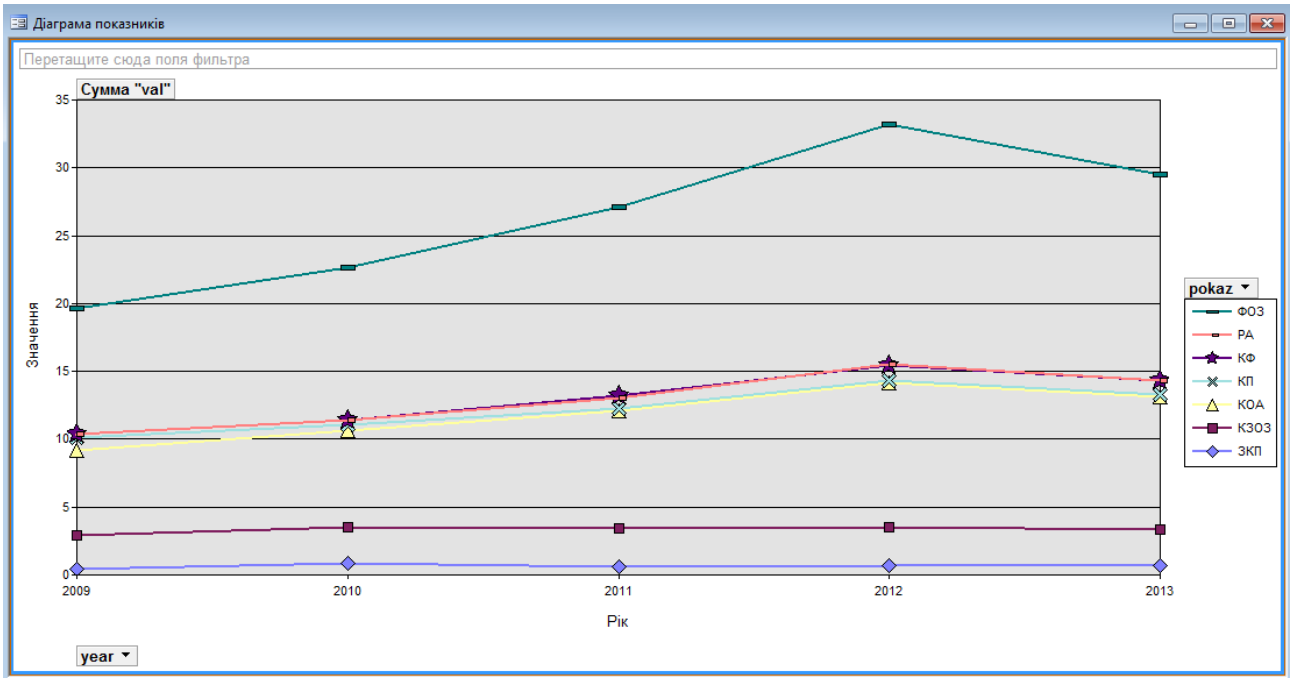


Рис. 4.19. Діаграма динаміки зміни показників

Як видно з діаграми, є тенденція до незначного росту значень показників. Тому є сенс далі досліджувати отримані часові ряди.

Далі на основі критерію Форстера-Стюарта шукаємо тренди. Отримуємо такі значення (таблиця 4.2):

Таблиця 4.2

Тренди даних показників регіону

Показник	**Закономірність**
КЗОЗ	КЗОЗ зростає
РА	РА спадає
ФОЗ	ФОЗ зростає
КОА	КОА зростає
КФ	КФ зростає
КП	КП спадає
ЗКП	ЗКП зростає

На основі табличних значень здійснюється автоматичний розрахунок показників. Результати розрахунку виводяться у файл Excel для забезпечення можливості їх подальшого аналізу (таблиця 4.3).

Таблиця 4.3

Розрахунок показників діагностики для прогнозування розвитку регіону

Рік	Коефіцієнт поточної (загальної) ліквідності	Коефіцієнт покриття	Коефіцієнт абсолютної ліквідності	Власні оборотні активи	Частка власних оборотних коштів у покритті запасів, %	Коефіцієнт забезпе-чення власними коштами	Коефіцієнт фінансової автономії	Коефіцієнт швидкої ліквідності	Чистий оборотний капітал	Коефіцієнт платоспро-можності (автономії)	Коефіцієнт фінансування	Коефіцієнт забезпеченості власними оборотними коштами	Коефіцієнт маневреності власного капіталу
2009		1,38		150,00	5,05		0,79	0,38	150,00	0,79		0,27	0,10
2010		0,97		-16,00	21,67		0,70	-0,03	-16,00	0,70		-0,03	-0,01
2011		0,61		-330,00	-39,10		0,51	-0,39	-330,00	0,51		-0,64	-0,38
2012		0,72		-276,00	-21,89		0,47	-0,28	-276,00	0,47		-0,38	-0,31
2013		0,59		-383,00	-47,01		0,49	-0,41	-383,00	0,49		-0,69	-0,43

Наступним етапом комплексного аналізу регіону є аналіз показників діагностики. З цією метою використано такі елементи інтерфейсу (рис. 4.20):

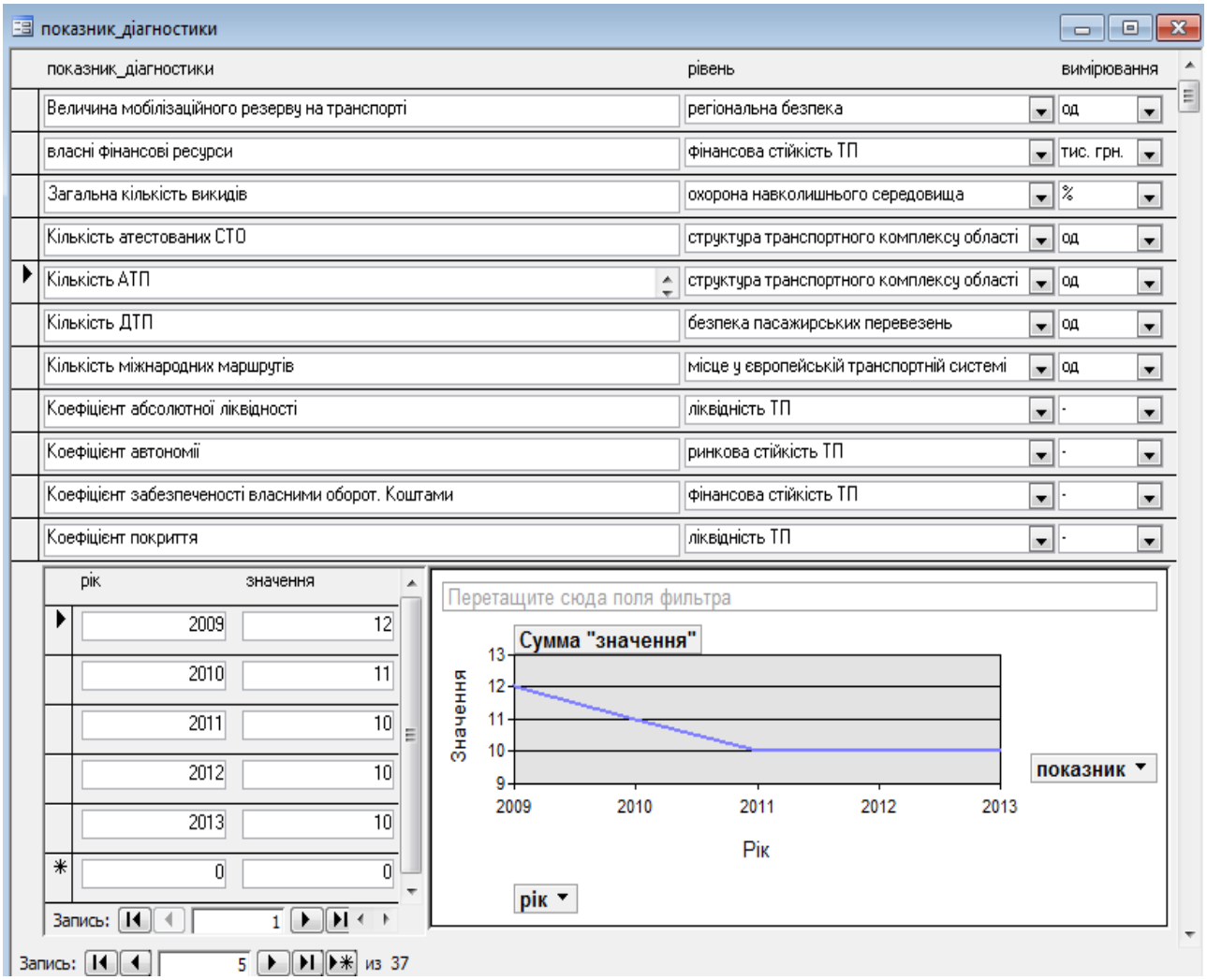


Рис. 4.20. Форма для аналізу показника діагностики

У нижній правій частині форми здійснюється автоматична побудова діаграми на онові значення показника за роками.

Нижче показано форма залежності для показника «Кількість міжнародних маршрутів» (рис. 4.21).

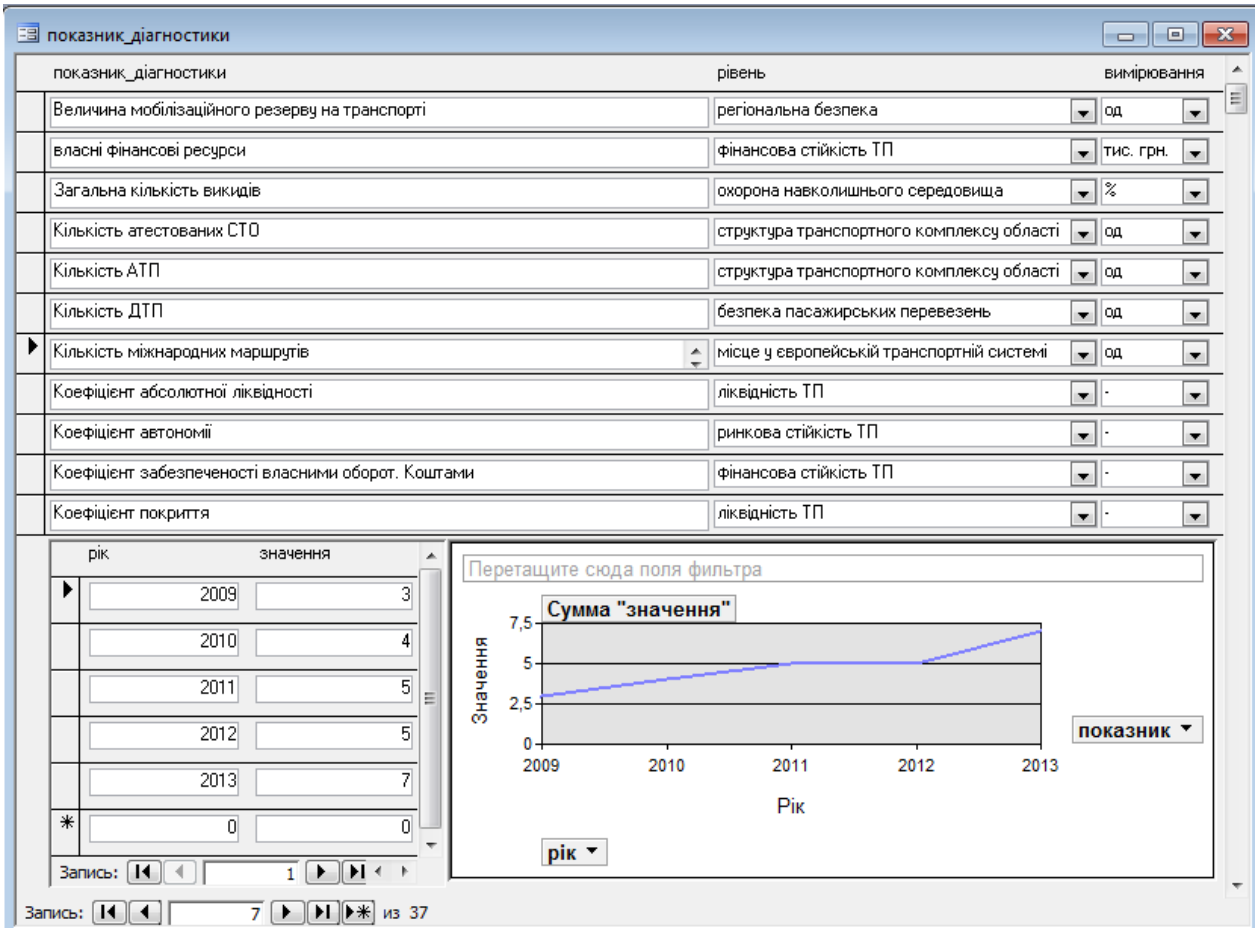


Рис. 4.21. Форма залежності для показника «Кількість міжнародних маршрутів»

Для визначення прогнозів за рівнями нормалізуємо значення показників за формулою:

$$x_{norm} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}. \qquad (4.2)$$

Отримаємо такі значення (таблиця 4.4.):

Таблиця 4.4

Нормалізовані показники діагностики

Нормалізовані_показники_діагностики			
Рівень	**min_max.показник_діагностики**	**val**	**рік**
матеріальні ресурси	необоротні активи	1	2009
матеріальні ресурси	необоротні активи	0,516908212560386	2010
матеріальні ресурси	необоротні активи	0,202898550724638	2011
матеріальні ресурси	необоротні активи	0	2012
матеріальні ресурси	необоротні активи	0,536231884057971	2013
фінансова стійкість ТП	власні фінансові ресурси	1	2009

Таблиця 4.4 (продовження)

фінансова стійкість ТП	власні фінансові ресурси	0,587596899224806	2010
фінансова стійкість ТП	власні фінансові ресурси	0	2011
фінансова стійкість ТП	власні фінансові ресурси	1,86046511627907E-02	2012
фінансова стійкість ТП	власні фінансові ресурси	2,48062015503876E-02	2013
інвестиції	рівень захисту інформації	0	2009
інвестиції	рівень захисту інформації	0,395348837209302	2010
інвестиції	рівень захисту інформації	0,511627906976744	2011
інвестиції	рівень захисту інформації	0,86046511627907	2012
інвестиції	рівень захисту інформації	1	2013
клієнти	обсяг реалізованих послуг	0	2009
клієнти	обсяг реалізованих послуг	0,147636674259681	2010
клієнти	обсяг реалізованих послуг	0,44376423690205	2011
клієнти	обсяг реалізованих послуг	1	2012
клієнти	обсяг реалізованих послуг	0,85378701594533	2013
клієнти	Собівартість перевезень	0,333333333333333	2009
клієнти	Собівартість перевезень	1	2010
клієнти	Собівартість перевезень	1	2011
клієнти	Собівартість перевезень	0	2012
клієнти	Собівартість перевезень	0,666666666666667	2013
клієнти	компенсація витрат по наданню послуг пільговим кат	1	2009
клієнти	компенсація витрат по наданню послуг пільговим кат	0,791666666666667	2010
клієнти	компенсація витрат по наданню послуг пільговим кат	0,708333333333334	2011
клієнти	компенсація витрат по наданню послуг пільговим кат	0,25	2012
клієнти	компенсація витрат по наданню послуг пільговим кат	0	2013
техніка	Частка транспортних засобів, що потребують заміни	1	2009
техніка	Частка транспортних засобів, що потребують заміни	0,75	2010
техніка	Частка транспортних засобів, що потребують заміни	0,5625	2011
техніка	Частка транспортних засобів, що потребують заміни	0,4375	2012
техніка	Частка транспортних засобів, що потребують заміни	0	2013
техніка	Частка нових транспортних засобів	0	2009
техніка	Частка нових транспортних засобів	0,25	2010
техніка	Частка нових транспортних засобів	0,4375	2011
техніка	Частка нових транспортних засобів	0,5625	2012
техніка	Частка нових транспортних засобів	1	2013
інвестиції	Обсяг інвестицій	1	2009
інвестиції	Обсяг інвестицій	1	2010
інвестиції	Обсяг інвестицій	1	2011
інвестиції	Обсяг інвестицій	1	2012
інвестиції	Обсяг інвестицій	1	2013
транспортна мережа	Міра покриття маршрутною мережею області	1	2009

Таблиця 4.4 (продовження)

транспортна мережа	Міра покриття маршрутною мережею області	1	2010
транспортна мережа	Міра покриття маршрутною мережею області	1	2011
транспортна мережа	Міра покриття маршрутною мережею області	1	2012
транспортна мережа	Міра покриття маршрутною мережею області	1	2013
транспортна мережа	Частка відремонтованих доріг	0	2009
транспортна мережа	Частка відремонтованих доріг	0,225806451612903	2010
транспортна мережа	Частка відремонтованих доріг	0,451612903225806	2011
транспортна мережа	Частка відремонтованих доріг	0,612903225806452	2012
транспортна мережа	Частка відремонтованих доріг	1	2013
транспортна мережа	Частка доріг, що потребують ремонту	1	2009
транспортна мережа	Частка доріг, що потребують ремонту	0,766666666666667	2010
транспортна мережа	Частка доріг, що потребують ремонту	0,533333333333333	2011
транспортна мережа	Частка доріг, що потребують ремонту	0,366666666666667	2012
транспортна мережа	Частка доріг, що потребують ремонту	0	2013
ринкова стійкість ТП	Коефіцієнт автономії	1	2009
ринкова стійкість ТП	Коефіцієнт автономії	0,666666666666666	2010
ринкова стійкість ТП	Коефіцієнт автономії	0	2011
ринкова стійкість ТП	Коефіцієнт автономії	0	2012
ринкова стійкість ТП	Коефіцієнт автономії	0	2013
регіональна безпека	Рентабельність перевезень	0,254901960784314	2009
регіональна безпека	Рентабельність перевезень	1	2010
регіональна безпека	Рентабельність перевезень	0	2011
регіональна безпека	Рентабельність перевезень	1	2012
регіональна безпека	Рентабельність перевезень	0,235294117647059	2013
регіональна безпека	Рентабельність активів	0,5	2009
регіональна безпека	Рентабельність активів	0,25	2010
регіональна безпека	Рентабельність активів	0	2011
регіональна безпека	Рентабельність активів	1	2012
регіональна безпека	Рентабельність активів	0,75	2013
ліквідність ТП	Коефіцієнт покриття	1	2009
ліквідність ТП	Коефіцієнт покриття	0,375	2010
ліквідність ТП	Коефіцієнт покриття	0	2011
ліквідність ТП	Коефіцієнт покриття	0,125	2012
ліквідність ТП	Коефіцієнт покриття	0	2013
ліквідність ТП	Коефіцієнт швидкої ліквідності	1	2009
ліквідність ТП	Коефіцієнт швидкої ліквідності	0,375	2010
ліквідність ТП	Коефіцієнт швидкої ліквідності	0	2011
ліквідність ТП	Коефіцієнт швидкої ліквідності	0,125	2012
ліквідність ТП	Коефіцієнт швидкої ліквідності	0	2013

Таблиця 4.4 (продовження)

ліквідність ТП	Коефіцієнт швидкої ліквідності	0	2013
ліквідність ТП	Коефіцієнт абсолютної ліквідності	0,784313725490196	2009
ліквідність ТП	Коефіцієнт абсолютної ліквідності	0,294117647058824	2010
ліквідність ТП	Коефіцієнт абсолютної ліквідності	0	2011
ліквідність ТП	Коефіцієнт абсолютної ліквідності	9,80392156862745E-02	2012
ліквідність ТП	Коефіцієнт абсолютної ліквідності	9,80392156862745E-02	2013
фінансова стійкість ТП	Коефіцієнт фінансування	1	2009
ліквідність ТП	Коефіцієнт абсолютної ліквідності	0,313725490196078	2010
ліквідність ТП	Коефіцієнт абсолютної ліквідності	0,843137254901961	2011
ліквідність ТП	Коефіцієнт абсолютної ліквідності	1	2012
ліквідність ТП	Коефіцієнт абсолютної ліквідності	0,931372549019608	2013
фінансова стійкість ТП	Коефіцієнт забезпеченості власними оборот. Коштами	0,706766917293233	2009
фінансова стійкість ТП	Коефіцієнт забезпеченості власними оборот. Коштами	0,255639097744361	2010
фінансова стійкість ТП	Коефіцієнт забезпеченості власними оборот. Коштами	0	2011
фінансова стійкість ТП	Коефіцієнт забезпеченості власними оборот. Коштами	0,195488721804511	2012
фінансова стійкість ТП	Коефіцієнт забезпеченості власними оборот. Коштами	1	2013
надходження до обласного бюджету від діяльності Тп	Частка надходжень до обл бюджету від діяльності ТП	1	2009
надходження до обласного бюджету від діяльності Тп	Частка надходжень до обл бюджету від діяльності ТП	1	2010
надходження до обласного бюджету від діяльності Тп	Частка надходжень до обл бюджету від діяльності ТП	1	2011
надходження до обласного бюджету від діяльності Тп	Частка надходжень до обл бюджету від діяльності ТП	1	2012
надходження до обласного бюджету від діяльності Тп	Частка надходжень до обл бюджету від діяльності ТП	1	2013
охорона навколишнього середовища	Загальна кількість викидів	1	2009
охорона навколишнього середовища	Загальна кількість викидів	0,666666666666667	2010
охорона навколишнього середовища	Загальна кількість викидів	0,333333333333333	2011
охорона навколишнього середовища	Загальна кількість викидів	0,666666666666667	2012
охорона навколишнього середовища	Загальна кількість викидів	0	2013
соціальний захист	Частка застрахованості пасажирів	1	2009
соціальний захист	Частка застрахованості пасажирів	1	2010
соціальний захист	Частка застрахованості пасажирів	1	2011
соціальний захист	Частка застрахованості пасажирів	1	2012
соціальний захист	Частка застрахованості пасажирів	1	2013
соціальний захист	Частка застрахованості працівників	0	2009

Таблиця 4.4 (продовження)

соціальний захист	Частка застрахованості працівників	0,153846153846154	2010
соціальний захист	Частка застрахованості працівників	0,384615384615385	2011
соціальний захист	Частка застрахованості працівників	0,692307692307692	2012
соціальний захист	Частка застрахованості працівників	1	2012
структура транспортного комплексу області	Кількість атестованих СТО	1	2009
структура транспортного комплексу області	Кількість атестованих СТО	0,785714285714286	2010
структура транспортного комплексу області	Кількість атестованих СТО	0,607142857142857	2011
структура транспортного комплексу області	Кількість атестованих СТО	7,14285714285714E-02	2012
структура транспортного комплексу області	Кількість атестованих СТО	0	2013
структура транспортного комплексу області	Кількість АТП	1	2009
структура транспортного комплексу області	Кількість АТП	0,5	2010
структура транспортного комплексу області	Кількість АТП	0	2011
структура транспортного комплексу області	Кількість АТП	0	2012
структура транспортного комплексу області	Кількість АТП	0	2013
управління транспортним комплексом	Співвідношк-сті прав і обов’язків органів, що управляють транспортом	1	2009
управління транспортним комплексом	Співвідношк-сті прав і обов’язків органів, що управляють транспортом	1	2010
управління транспортним комплексом	Співвідношк-сті прав і обов’язків органів, що управляють транспортом	1	2011
управління транспортним комплексом	Співвідношк-сті прав і обов’язків органів, що управляють транспортом	1	2012
управління транспортним комплексом	Співвідношк-сті прав і обов’язків органів, що управляють транспортом	1	2013

Таблиця 4.4 (продовження)

управління транспортним комплексом	Повнота нормативної бази по управлінню транспортом	0	2009
управління транспортним комплексом	Повнота нормативної бази по управлінню транспортом	0	2010
управління транспортним комплексом	Повнота нормативної бази по управлінню транспортом	0,222222222222222	2011
управління транспортним комплексом	Повнота нормативної бази по управлінню транспортом	0,722222222222222	2012
управління транспортним комплексом	Повнота нормативної бази по управлінню транспортом	1	2013
місце у європейській транспортній системі	Кількість міжнародних маршрутів	0	2009
місце у європейській транспортній системі	Кількість міжнародних маршрутів	0,25	2010
місце у європейській транспортній системі	Кількість міжнародних маршрутів	0,5	2011
місце у європейській транспортній системі	Кількість міжнародних маршрутів	0,5	2012
місце у європейській транспортній системі	Кількість міжнародних маршрутів	1	2013
місце у європейській транспортній системі	Міра відповідності транспортного комплексу області міжнародним стандартам	0	2009
місце у європейській транспортній системі	Міра відповідності транспортного комплексу області міжнародним стандартам	0,307692307692308	2010
місце у європейській транспортній системі	Міра відповідності транспортного комплексу області міжнародним стандартам	0,641025641025641	2011
місце у європейській транспортній системі	Міра відповідності транспортного комплексу області міжнародним стандартам	0,923076923076923	2012
місце у європейській транспортній системі	Міра відповідності транспортного комплексу області міжнародним стандартам	1	2013
місце у європейській транспортній системі	Наявність міжнародних транспортних коридорів	1	2009
місце у європейській транспортній системі	Наявність міжнародних транспортних коридорів	1	2010

Таблиця 4.4 (продовження)

місце у європейській транспортній системі	Наявність міжнародних транспортних коридорів	1	2011
місце у європейській транспортній системі	Наявність міжнародних транспортних коридорів	1	2012
місце у європейській транспортній системі	Наявність міжнародних транспортних коридорів	1	2013

Далі здійснимо згортку показників мультиплікативним методом. Для цього перемножимо значення нормованих показників за рівнями та за роками, оскільки маємо нормовані показники у різних одиницях виміру. Після цього знайдемо тренд рівня на основі критерію Форстера-Стюарта (таблиця 4.5):

Таблиця 4.5

Тренд рівня показника діагностики «Кількість міжнародних маршрутів»

Тренд_показник_діагностики	
Рівень	**тренд**
Інвестиції	інвестиції зростає
Клієнти	клієнти спадає
ліквідність ТП	ліквідність ТП спадає
матеріальні ресурси	матеріальні ресурси спадає
місце у європейській транспортній системі	місце у європейській транспортній системі спадає
надходження до обласного бюджету від діяльності Тп	надходження до обласного бюджету від діяльності Тп зростає
охорона навколишнього середовища	охорона навколишнього середовища спадає
регіональна безпека	регіональна безпека спадає
ринкова стійкість ТП	ринкова стійкість ТП спадає
соціальний захист	соціальний захист спадає
структура транспортного комплексу області	структура транспортного комплексу області спадає
Техніка	техніка зростає
транспортна мережа	транспортна мережа спадає
управління транспортним комплексом	управління транспортним комплексом спадає
фінансова стійкість ТП	фінансова стійкість ТП спадає

Прогноз показників здійснюється з допомогою регресійного аналізу шляхом перебору моделей для пошуку найбільшого коефіцієнту кореляції. Аналізуються такі типи регресії:

- лінійна,
- логарифмічна,
- степенева,
- з квадратним коренем.

У якості параметрів регресії використовуються:

- змінна X – рік,
- змінна У – прогнозовані коефіцієнти.

На рисунку 4.22 показано результат прогнозу на 2014 рік з допомогою визначення лінійної регресії.

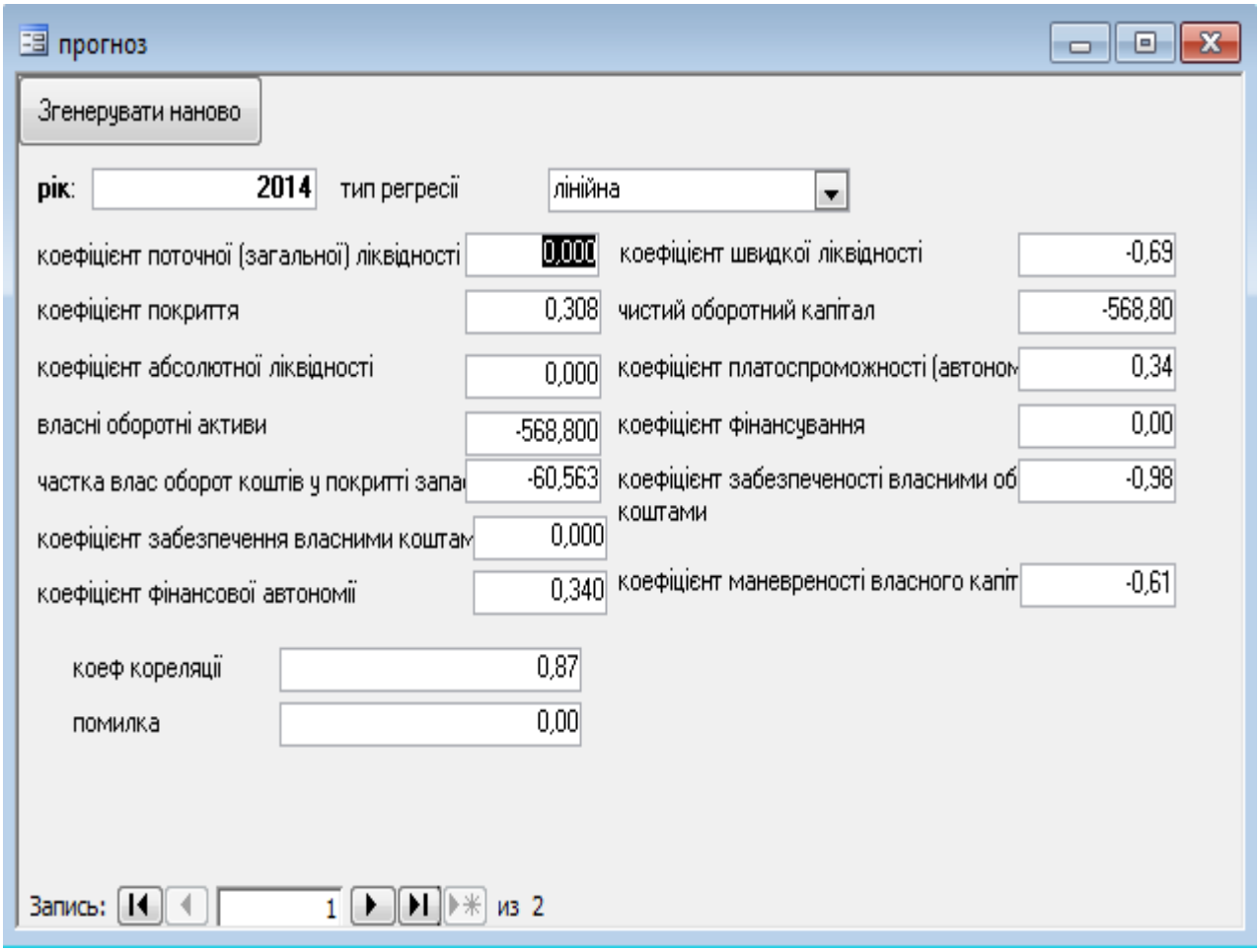


Рис. 4.22. Форма результатів прогнозу на 2014 рік

Обрано лінійну модель як модель з найбільшим коефіцієнтом кореляції.

Адекватність моделі можна оцінити, спрогнозувавши параметри за відомий рік, наприклад, за 2013 рік (рис. 4.23). Отримано такий результат:

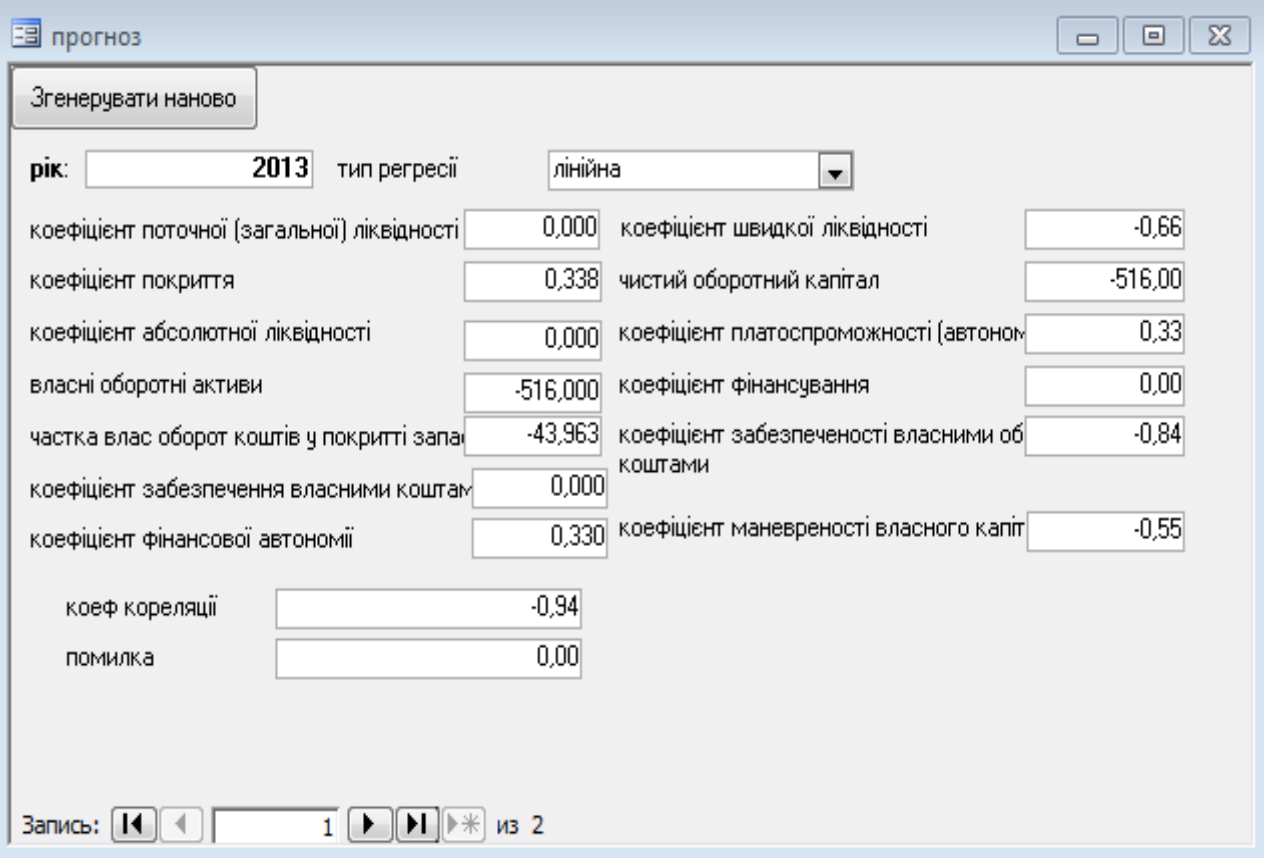


Рис. 4.23. Форма результатів прогнозу на 2013 рік

У таблиці 4.6 наведено реальні значення показників діагностики за 2013 рік.

Таблиця 4.6

Реальні значення показників діагностики за 2013 рік

Показник	Алгоритм розрахунку	Нормативне значення	2013р
коефіцієнт поточної (загальної) ліквідності	(ОА+ВМП) / ПЗ	≥1,5	0,6
коефіцієнт покриття	ОА/ПЗ	>1	0,6
коефіцієнт абсолютної ліквідності (платоспроможності)	(ГК+ПФІ)/ПЗ	0,1-0,2	0,2
власні оборотні активи	ВК-НА	>0	-383
частка власних оборотних коштів у покритті запасів, %	ВОАх100/З	>50%	1532

Таблиця 4.6 (продовження)

коефіцієнт забезпечення власними коштами	(ВК+ЗНВП+ДМП-НА) /ОА	≥0,1	-0,69
коефіцієнт фінансової автономії	ВК/Б	>0,5	0,5
коефіцієнт швидкої ліквідності	(ОА-З) /ПЗ	0,6-0,8	0,6
коефіцієнт абсолютної ліквідності	(ГК+ПФІ)/ПЗ	>0	0,2
чистий оборотний капітал	ОА-ПЗ	>0	-383
коефіцієнт платоспроможності (автономії)	ВК/Б	>0,5	0,5
коефіцієнт фінансування	(ЗНВП+ДЗ+ПЗ+ДМП)/ВК	<1	1,05
коефіцієнт забезпеченості власними оборотними коштами	(ОА-ПЗ) /ОА	>0,1	-0,69
коефіцієнт маневреності власного капіталу	(ОА-ПЗ) /ВК	>0	-0,4

Існуюча невелика розбіжність виникає у зв’язку з малою кількістю рівнів (лише 4 роки спостережень).

Далі визначено силу зв’язку між параметрами. Це здійснено на основі розрахунку коефіцієнту кореляції. Значення сили зв’язку подається лінгвістичною змінною з такими межами (таблиця 4.7):

Таблиця 4.7

Значення сили зв’язку для розрахунку коефіцієнта кореляції

№	Назва змінної	Межі
1	Зв’язок відсутній	0 – 0,4
2	Слабкий зв’язок	0,41 – 0,7
3	Значний зв’язок	0,5 – 0,69
4	Сильний зв’язок	0,7 – 0,89
5	Дуже сильний зв’язок	0,9 – 0,99
6	Функціональний зв’язок	1

До уваги береться пара тих параметрів, зв’язок між якими є не меншим, ніж «значний зв’язок». Отримано такі результати аналізу:

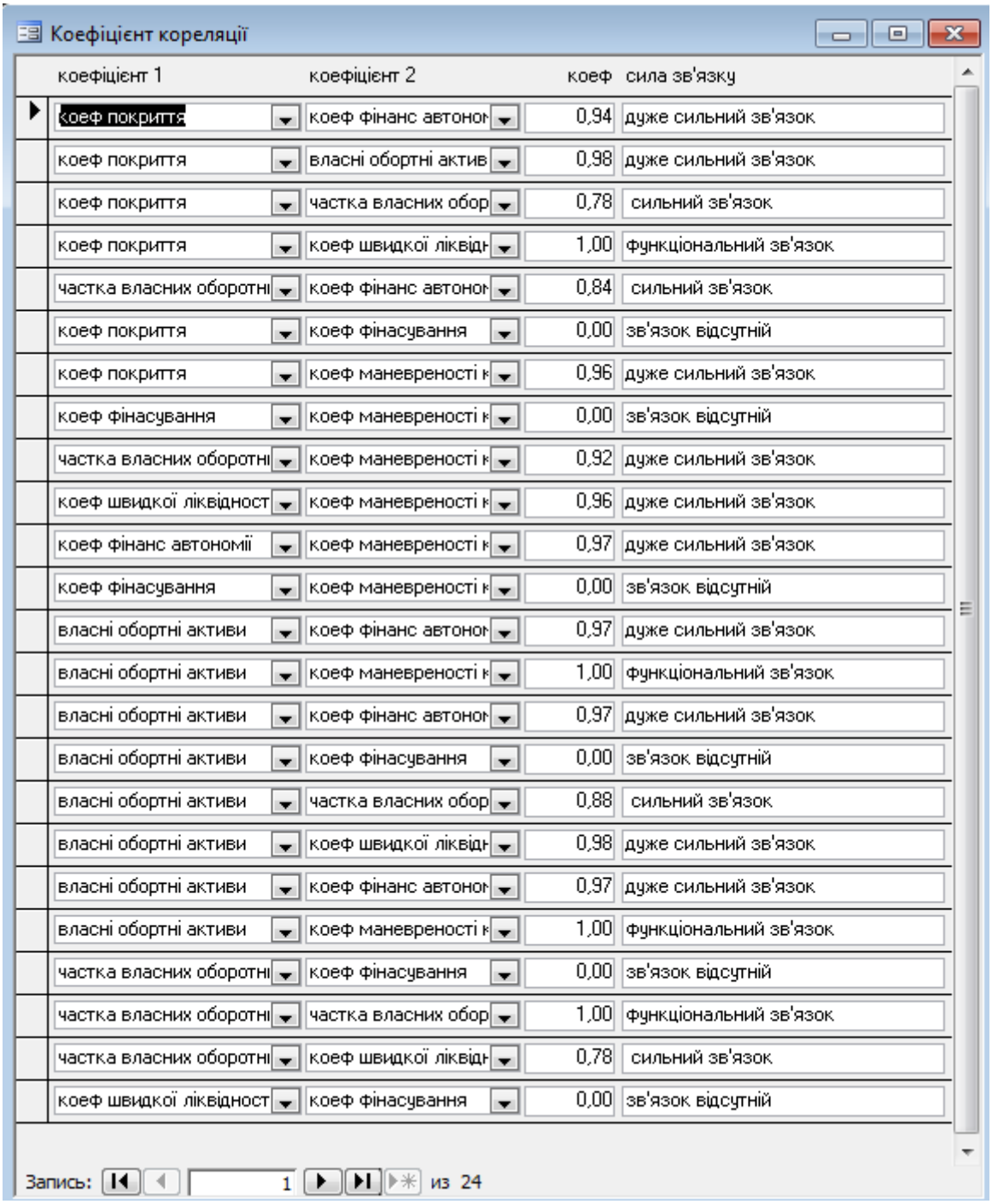


Рис. 4.24. Значення коефіцієнтів кореляції

Наявність сильного зв’язку між деякими параметрами очевидна. Наприклад, значення параметра «Власні оборотні активи» та «Коефіцієнт фінансової автономії» прямо пропорційні значенню показника «Власний показник». Залежність показників «Коефіцієнт покриття» та «Коефіцієнт

фінансової автономії» також є лінійною, і вони обоє залежать від оборотності активів. Це означає, що у разі розв’язання багатокритеріальної задачі максимізації суми показників критеріїв з деякими коефіцієнтами достатньо працювати лише над збільшенням певних параметрів забезпечення наступних витрат і платежів, і це автоматично призведе до збільшення значення інших показників.

Знаходження параметрів, між якими встановлено функціональний зв’язок, вказує на те, що у задачі багатокритеріальної оптимізації один з параметрів пари може бути опущений, що суттєво зменшує кількість критеріїв.

Висновки до розділу 4

1. Спроектовано архітектуру інформаційної системи на основі Великих даних, в якій враховані різні рівні подання даних та модулі перетворення інформації з джерел даних в модель «сутність-характеристика», а також відповідь на запит користувача, що дає змогу звертатися лише до необхідних джерел, а не завантажувати всю інформацію.
2. Побудовано схему сховища даних регіону для зберігання як детальних, так і агрегованих показників.
3. Побудовано алгоритм отримання даних з різних джерел та їх аналізу.
4. Розроблено засоби для розрахунку та аналізу основних показників розвитку регіону.

Матеріали розділу опубліковано у [8, 12].

ВИСНОВКИ

У дисертаційній роботі розв’язано важливе наукове завдання розроблення методів та засобів організації та інтеграції інформаційних ресурсів Великих даних для процесів управління регіоном. У результаті виконання цієї роботи одержано такі результати.

1. Проаналізовано методи, моделі та засоби опрацювання різнотипних даних, інформаційних технологій роботи з Великими даними. Обґрунтовано актуальність розв’язання завдання організації та інтеграції інформаційних ресурсів Великих даних у системах територіального управління.
2. Розроблено інформаційну модель Великих даних «сутність-характеристика», яка дає змогу організовувати структуровані та слабоструктуровані дані і на відміну від багатовимірної моделі не містить надлишковості.
3. Розроблено методи перетворення реляційних та слабоструктурованих даних у дані, подані в моделі «сутність-характеристика», що дало змогу уніфікувати форму запитів до різних моделей даних.
4. Розроблено федеративний метод інтеграції даних через визначення пари «сутність-характеристика» та узгодження семантики, що на відміну від методів інтеграції даних на рівні сховища даних дозволило інтегрувати дані з джерел з наперед невідомою структурою даних без попереднього локального завантаження, і що, в свою чергу, дало змогу підвищити ефективність подальшого аналізу Великих даних. Цей метод став основою для розроблення методу формування відповіді на запит користувача.
5. Розроблено метод формування відповіді на запит користувача до Великих даних шляхом уніфікації елементів мов запитів до різних інформаційних джерел, що дало змогу трансформувати запит до Великих даних у запит на основі ключових слів.
6. Розроблено програмні компоненти інформаційної системи підтримки прийняття рішень «Інтегратор» для управління регіоном з використанням Великих даних.

СПИСОК ЛІТЕРАТУРИ

1. Болюбаш Ю. Я. Методи та засоби опрацювання великих даних у системах територіального управління / Ю. Я. Болюбаш // Науковий вісник Національного лісотехнічного університету: збірник наукових праць. – Львів : РВВ НЛТУ України. – 2016. – Вип. 26.4. – С. 341–354.
2. Болюбаш Ю. Я. Методи опрацювання Великих даних у федеративному сховищі даних / Ю. Я. Болюбаш // Вісник Національного університету «Львівська політехніка». – 2016. – № 843: Комп'ютерні науки та інформаційні технології. – С. 356–365.
3. Big Data information technology and data space architecture / N. Shakhovska, O. Veres, Y. Bolubash, L. Bychkovska // Sensors & Transducers. – 2015. – Vol. 195, Is.12. – P. 69–77.
4. Шаховська Н. Б. Опрацювання невизначеності у великих даних / Н. Б. Шаховська, Ю. Я. Болюбаш // Радіоелектроніка, інформатика, управління. – 2014. – № 1. – С. 96–105.
5. Шаховська Н. Б. Організація великих даних у розподіленому середовищі / Н. Б. Шаховська, Ю. Я. Болюбаш, О. М. Верес // Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація. – 2014. –№ 2. – С. 147–155.
6. Шаховська Н. Б. Робота з великими даними – показниками соціо-еколого-економічного розвитку регіону / Н. Б. Шаховська, Ю. Я. Болюбаш // Восточно-Европейский журнал передовых технологий. – 2013. – № 5(2). – С. 4–8.
7. Шаховська Н. Б. Модель Великих даних «сутність-характеристика» / Шаховська Н. Б., Болюбаш Ю. Я. // Вісник Національного університету «Львівська політехніка». – 2015. – № 814 : Інформаційні системи та мережі. – С. 186–196.
8. Shakhovska N. Dataspace architecture and manage its components class projection / N. Shakhovska, Y. Bolubash // Econtechmod / Polish Academy of Sciences, Branch in Lublin. – 2015. – Vol. 4, №. 1. – P. 89–97.

9. Shakhovska N. Big Data model «entity and characteristics» / N. Shakhovska, Y. Bolubash // Econtechmod / Polish Academy of Sciences, Branch in Lublin. – 2015. – Vol. 4, No. 2. – P. 51–58.

10. Шаховська Н. Б. Методи прогнозування соціо-еколого-економічного розвитку регіону Західного Бугу / Н. Б. Шаховська, Ю. Я. Болюбаш // Матеріали IX Міжнародної науково-практичної конференції «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту» (ISDMCI), 20-24 травня 2013 р., Євпаторія. – Херсон, 2013. – С. 324–326.

11. Шаховська Н. Б. Федеративне сховище даних для Big Data / Н. Б. Шаховська, Ю. Я. Болюбаш // Математика. Інформаційні технології. Освіта: матеріали Міжнародної науково-практичної конференції, 6-8 червня 2014 р., Луцьк; Світязь. – Луцьк, 2014. – С. 87–88.

12. Шаховська Н. Б. Проектування федеративного сховища даних на основі Великих даних / Н. Б. Шаховська, Ю. Я. Болюбаш // Матеріали III Міжнародної науково-практичної конференції Інформаційні управляючі системи та технології (ІУСТ), 23-25 вересня 2014 р., Одеса. – Одеса, 2014. – С. 293–296.

13. Шаховська Н. Б. Робота з великими даними – показниками соціо-еколого-економічного розвитку регіону / Н. Б. Шаховська, Ю. Я. Болюбаш // Математика. Інформаційні технології. Освіта: матеріали II Міжнародної науково-практичної конференції, 3-5 червня 2013 р., Луцьк-Світязь. – Луцьк, 2013. – С. 41–42.

14. Data Space architecture for Big Data managering / Shakhovska Natalya, Veres Oleh, Bolubash Yuri, Liliana Bychkovska // Proceedings of the International Conference on Computer Sciences and Information Technologies, 14-17 September 2015, Lviv. – Lviv, 2015. – P. 184–187.

15. Shakhovska N. B. Big Data federated repository model / N. B. Shakhovska, Y. J. Bolubash, O. M. Veres // Proceedings of 13th International Conference: The Experience of Designing and Application of CAD Systems in Microelectronics, CADSM, 24-27 February 2015, Lviv. – Lviv, 2015. – P. 382–384.

16. Горчаков, А. Математичний апарат для інвестора. Аналіз та прогнозування часових рядів [Текст] / А. Горчаков // Фінансовий ринок України. – 2007. – 12. – С. 38–42.

17. Кісь Я.П. Інтелектуальні геоінформаційні системи. Міжнародний досвід та шляхи розвитку в Україні [Текст] / Я.П.Кісь, Н.Б.Шаховська, О.Б.Вальчук // Вісник Національного університету «Львівська політехніка». – Л.: Вид-во Нац. ун-ту «Львів. політехніка», 2008. – №653: Інформаційні системи та мережі. – С. 139–145.

18. Шаховська Н.Б. Формальне подання простору даних у вигляді алгебраїчної системи [Текст] / Шаховська Н.Б. // Системні дослідження та інформаційні технології (System research & information technologies: міжнародний науково-технічний журнал) / Національна академія наук України, Інститут прикладного системного аналізу. – Київ, 2011. – № 2. – С. 128 – 140.

19. Лычкина Н.Н. Компьютерное моделирование социально-экономического развития регионов в системах поддержки принятия решений [Текст] / Н.Н. Лычкина // Материалы III Международной конференции «Идентификация систем и задачи управления» SICPRO04, 28-30 января 2004, Москва. – М.: ИПУ РАН, 2004. – С. 1377–1402.

20. White Tom Hadoop: The Definitive Guide [Електронний ресурс]. – Режим доступу: http://download.bigbata.com/ ebook/oreilly/books/ Hadoop.The. Definitive.Guide.3rd.Edition.May.2012.pdf.

21. Hilbert M. Big Data for Development: From Information- to Knowledge Societies [Electronic Resours] / Martin Hilbert. Access mode: http://papers. ssrn.com/abstract=2205145.

22. Шаховська Н.Б. Формальне подання простору даних у вигляді алгебраїчної системи [Текст] / Шаховська Н.Б. // Системні дослідження та інформаційні технології / Національна академія наук України, Інститут прикладного системного аналізу. – Київ, 2011. – № 2. – С.128 – 140.

23. Згуровський М.З. Основи системного аналізу [Текст] / Згуровський М.З., Панкратова Н.Д. – К.: Видавнича група BHV, 2007. – 544 с.: іл.

24. Акатова Н.А. Комплексность проектирования интегрированных информационных систем административно-территориального и муниципального управления [Текст] / Акатова Н.А., Черкасов Ю.М. // Вестник Университета: серия «Информационные системы управления». – М.: ГУУ, 2000. – № 1. – С.23-35.

25. Exploiting technical terminology for knowledge management / Rinaldi F, Yuste E., Schneider G., Hess M., Roussel D. // Proceedings of ECAI and EKAW, 2004. – Режим доступу: http://ceur-ws.org/Vol-121/02.pdf.

26. Литвин В.В. Методи та засоби інженерії даних та знань / В.В.Литвин – Львів: Магнолія-2006, 2012. – 241 с.

27. Information Granularity, Big Data, and Computational Intelligence [Електронний ресурс] / W. Pedrycz and S.-M. Chen (eds.). – Access mode: https://books.google.com.ua/books?id.

28. Big data analytics // Proceedings of the First International Conference on Big Data Analytics BDA'2012. Lecture Notes in Computer Science [Text] / Srinivasa, S., Bhatnagar, V. (eds.), 24–26 Dec 2012, New Delhi. – Vol. 7678. – New Delhi: Springer, 2012. – P. 1–179.

29. Бутакова М.А. Мера информационного подобия для анализа слабоструктурированной информации [Електронний ресурс] / Бутакова М.А., Климанская Е.В., Янц В.И. // Современные проблемы науки и образования. – 2013. – № 6. – Режим доступу: http://www.science-educa- tion.ru/113-11307.

30. Bigtable: A Distributed Storage System for Structured Data [Electronic Resours] / Chang Fay, Dean Jeffrey, Ghemawat Sanjay, Hsieh Wilson C., Wallach, Deborah A., Burrows Michael, Chandra Tushar, Fikes Andrew, Gruber Robert E. Access mode: http://static.googleusercontent.com/media/research.google.com/ru// archive/bigtable-osdi06.pdf.

31. Papakonstantinou Y. Object exchange across heterogeneous information sources [Text] / Y. Papakonstantinov, FI. Garcia-Molina, J. Widom // Proc.

of 11-th International Conference on Data Engineering (ICDE'05), April 2005, Tokyo, Japan. – Tokyo, 2005. – P. 251 - 261.

32. Zhou Feng Efficient pattern discovery for semistructured data / Zhou Feng, W. Hsu, Mong Li Lee. // Proc. of 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-05), 14-16 Nov. 2005, Hon Kong. - Hon Kong, 2005. - P. 301-309.
33. Шаховська Н.Б. Програмне та алгоритмічне забезпечення сховищ та просторів даних : монографія [Текст] / Н.Б. Шаховська. – Львів : Вид-во Львівської політехніки, 2010.– 194 с.
34. Шаховська Н. Особливості моделювання просторів даних [Текст] / Н. Шаховська // Вісник Національного університету «Львівська політехніка». – Л. : Вид-во Нац. ун-ту «Львівська політехніка», 2008. – № 608 : Комп'ютерна інженерія та інформаційні технології. – С. 145 – 154.
35. Shakhovska N. Application of algorithms of classification for uncertainty reduction [Text] / Natalya Shakhovska, Mykola Medykovsky, Petro Stakhiv // Przeglad Elektrotechniczny. – 2013. – Vol 89, №4. – P. 284–286.
36. Кут В.І. Модель консолідованих даних дистанційного навчально-консультаційного центру осіб із особливими потребами [Текст] / В.І. Кут // Вісник Національного університету «Львівська політехніка». – 2014. – №783: Інформаційні системи та мережі. – С. 120–127.
37. Кушнірецька І. І. Аналіз інформаційних ресурсів системи динамічної інтеграції слабоструктурованих даних у Web-середовищі [Текст] / І. І. Кушнірецька, О. І. Кушнірецька, А. Ю. Берко // Вісник Національного університету «Львівська політехніка». – 2014. – № 805: Інформаційні системи та мережі. – С. 162–169.
38. Magoulas Roger Big data: Technologies and techniques for large scale data [Electronic Resours] / Roger Magoulas, and Lorica Ben. – Access mode: http://assets.en.oreilly.com/1/event/54/mdw_online_bigdata_radar_pdf.pdf.

39. Kossmann D. Personal Data Spaces [Electronic Resours] / D. Kossmann, J.P. Dittrich. – Access mode: http://www.inf.ethz.ch/news/focus/res_focus/feb_2006/index_DE.

40. Hooman J. Equivalent semantic models for a distributed Data Space architecture [Текст] / J. Hooman, J.van de Pol // Formal Methods for Components and Objects. – Berlin; Heidelberg: Springer, 2003. – P. 182-201.

41. The Open Archives Initiative Protocol for Metadata Harvesting Protocol Version 2.0 of 2002-06-14. [Electronic Resours]. – Access mode: http://www.openarchives.org/OAI/2.0 /openarchivesprotocol.htm.

42. Gritsenko V.I. Information technologies, trends, ways of development [Text] / V.I.Gritsenko, and A.A.Ursatev // Control systems and machines. – 2001. – № 5. – P. 3–20.

43. Литвин В.В. Метод моделювання процесу підтримки прийняття рішень у конкурентному середовищі / В.В.Литвин, О.В.Оборська, Р.В. Вовнянка // Математичні машини й системи. – Київ, 2014. – №1. – С. 50-57.

44. Свами М. Графы, сети и алгоритмы / М. Свами, К. Тхуласираман. – М.: Наука, 1984. – 256 с.

45. IBM What is big data? – Bringing big data to the enterprise [Electronic Resours]. – Access mode: www.ibm.com.

46. Oracle and FSN: Mastering Big Data: CFO Strategies to Transform Insight into Opportunity [Electronic Resours]. Access mode: http://www.oracle.com/us/solutions/ent-performance-bi/business-intelligence/mastering-big-data- cfo-strategies-1853061.pdf.

47. Jacobs A. The Pathologies of Big Data [Text] / Jacobs A. // Databases. – 2009. – Vol. 7, issue 6. – P.1-12.

48. Magoulas R. Introduction to Big Data [Electronic Resours] / R. Magoulas, B.Lorica. – Access mode: http://www.oreilly.com/data/free/release-2-issue-11.csp.

49. Snijders C. «Big Data»: Big gaps of knowledge in the field of Internet [Electronic Resours] / C.Snijders, U.Matzat, and U.-D.Reips // International Journal of Internet Science. – Vol.7. – P. 1-5. – Access mode: http://www.ijis.net/ijis7_1/ijis7_1_editorial.html.

50. Laney D. 3D Data Management: Controlling Data Volume, Velocity and Variety [Electronic Resours] / D.Laney. – Access mode: https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data.

51. Gartner Says Solving «Big Data» Challenge Involves More Than Just Managing Volumes of Data [Electronic Resours]. – Access mode: http://www.gartner.com/newsroom/id/1731916.

52. Laney D. The Importance of «Big Data»: A Definition [Text] [Electronic Resours] / Mark A. Beyer, Douglas Laney. – Access mode: https://www.gartner.com/doc/2057415/importance-big-data-definition.

53. National Research Council. Behavioral Modeling and Simulation: From Individuals to Societies, Committee on Organizational Modeling: From Individuals to Societies / G. L. Zacharias, J. MacMillan, S. B. Van Heme l (eds.); Board on Behavioral, Cognitive, Sensory Sciences, Division of Behavioral, Social Sciences and Education [Text]. – Washington: The National Academies Press, 2008.

54. MapReduce and Parallel DBMSs: Friends or Foes [Text] / Stonebraker M., Abadi D., DeWitt D. J., Madden S., Pavlo A., Rasin, A. // Communications of the ACM. – 2012. – Vol. 53, №1. – P. 64-71.

55. «Big Data» [Електронний ресурс]. Режим доступу: http://www.tadviser.ru/index.php/Статья:Большие_данные_%28Big_Data%29#cite_note-g-6 .

56. Ohlhorst Frank J. A Cloudy Year for Big Data. eWeek [Electronic Resours] / Frank J. Ohlhorst. – Access mode: http://www.eweek.com/c/a/ Cloud-Computing/2012-A-Cloudy-Year-for-Big-Data-102807.

57. Chernyak L. Big Data - the new theory and practice. Open the system [Electronic Resours] / Leonid Chernyak. – M. : Open Systems, 2011. – № 10. – Access mode: http://www.osp.ru/os/2011/ 10/13010990.

58. «Big Data» Brighten BI Future. eWeek [Electronic Resours]. – Access mode: http://www.eweek.com/c/a/Data-Storage/TBA-Hadoop-Yahoo-Big-Data-Brightens-BI-Future-254079.

59. Gartner Says Solving «Big Data» Challenge Involves More Than Just Managing Volumes of Data. [Electronic Resours]. – Access mode: http://www.gartner.com/newsroom/id/1731916.

60. Christensen Clayton M. The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail [Text] / Clayton M. Christensen. – Harvard: Business School Press, 1997.

61. Correlation does not imply causation [Electronic Resours]. – Access mode: http://en.wikipedia.org/wiki/Correlation_does_not_imply_causation (Accessed 5/5/13).

62. Cyber-Physical System Wikipedia page [Electronic Resours]. – Access mode: http://en.wikipedia.org/wiki/Cyber-physical_system (Accessed 5/5/13).

63. Гендель Е. Г. Применение алгебраических моделей для синтеза процессов обработки файлов [Текст] / Гендель Е. Г., Мунерман В. И. // Управляющие системы и машины. – Киев: Наукова думка, 1984. – № 4. – С.32–39.

64. Глущенко В.В. Разработка управленческих решений. Прогнозирование – планирование, теория проектирования экспериментов [Текст] / Глущенко В.В., Глущенко И.И. – М.: Крылья, 1997. – 308 с.

65. ГОСТ 28195-89. Оценка качества программных средств. Общие положення [Електронний ресурс]. – Режим доступу: http://www.complexdoc.ru/text/ГОСТ%2028195-89.

66. Гриценко В.И. Информационные технологи: тенденции, пути развития [Текст] / В.И.Гриценко, А.А.Урсатьев // Управляющие системы и машины. – 2001. – № 5. – С. 3–20.

67. Додонов С.Б. О системе обработки нечеткой и неполной информации для поддержки принятия решения в задаче противоборства [Текст] / Додонов С.Б., Квачев В.Г., Петрушенко Л.А. // Управляющие системы и машины. – Москва, 2001. – Январь-февраль (№ 1). – С. 90–92.

68. Документация по SQL Server 2005 [Електронний ресурс]. – Режим доступу: http://msdn.microsoft.com/ru-ru/library/ms203721(SQL.90).aspx.

69. Драган Я.П. Описание тональных кардиосигналов с помощью модели периодически коррелированных случайных процессов [Текст] / Я.П. Драган, Г.М.Осухивская // Проблемы управления и информатики. – 1999. – № 1. – С. 78–84.

70. Драган Я.П. Структура и представления моделей стохастических сигналов [Текст] / Драган Я.П. – Киев: Наук.думка, 1980. – 384 с.

71. Драган Я.П. Методы вероятностного анализа ритмики океанологических процессов [Текст] / Драган Я.П., Рожков В.А., Яворский И.Н. – Л.: Гидрометеоиздат, 1987. – 320 с.

72. Дружинин В.В. Введение в теорию конфликта [Текст] / Конторов Д.С., Конторов М. Д. – М.: Радио и связь, 1989. – 213 с.

73. ДСТУ ISO 9001-2001. Системи управління якістю. Вимоги. – Київ: Держстандарт України, 2001. – 29 с.

74. Дюбуа А. Теория вероятностей. Приложение к представлению знаний в информатике [Текст] / Дюбуа А., Прад А. – Москва, «Радио и связь». – 1990. – 320 с.

75. Елтаренко Е. Оценка аппаратных и программных средств по многоуровневой системе критериев [Текст] / Елтаренко Е., Сергиевский М. // Компьютер-пресс. – 1998. – №8. – С. 268–272.

76. Жежнич П.І. Семантично відкриті інформаційні системи [Текст] / Жежнич П.І., Кравець Р.Б., Пасічник В.В., Пелещишин А.М. // Вісник Державного університету «Львівська політехніка». – Л.: Вид-во Нац. ун-ту «Львівська політехніка», 1999. – № 383: Інформаційні системи та мережі. – С. 84–96.

77. Жежнич П.І. Методи подання та опрацювання невизначеностей для систем навчання [Текст] / Жежнич П.І., Шаховська Н.Б. // Вісник Національного університету «Львівська політехніка». – Л.: Вид-во Нац. ун-ту «Львівська політехніка», 2006. – № 565: Комп'ютерні науки та інформаційні технології. – С. 275–282.

78. Жижимов О.Л. Введение в Z39.50 [Електронний ресурс] / Жижимов О.Л. – Новосибирск, 2003. – Режим доступу: http://z3950.uig-gm.nsc.ru:210/introduction/Part_tit.htm.

79. Заде Л. Понятие лингвистической переменной и его применение к принятию приближенных решений [Текст] / Заде Л. – М.: Мир, 1976. – 166 с.

80. Конструирование канонических информационных моделей для интегрированных информационных систем [Текст] / Захаров В. Н., Калиниченко Л. А., Соколов И. А., Ступников С. А. // Информатика и ее применения. – М., 2007. – Т. 1, Вып. 2. – С. 67–71.

81. Зиновьев П.А. Вопросы теории и практики создания и развития корпоративных систем в отрасли связи [Текст] / Зиновьев П.А, Насыров И.З. // Открытые системы. – 2006. – № 3. – С. 18–22.

82. Инициатива SDMX: новые подходы к обмену статистическими данными [Електронний ресурс]. Режим доступу: http://citforum.univ. kiev.ua/internet/xml/sdmx/.

83. Инмон Б. Производительность систем хранилищ данных [Текст] / Инмон Б. // Performance In The Data Warehouse Environment. – 2000. – №4. – С. 41–48.

84. Інформаційне суспільство [Електронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/Інформаційне_суспільство.

85. Калиниченко Л.А. Методология организации решения задач над множественными распределенными неоднородными источниками информации [Текст] / Калиниченко Л.А. // Сборник трудов Международной конференции «Современные информационные технологии и ИТ-образование» – М.: МГУ, 2005. – С. 20–37.

86. Калиниченко Л.А. Методы и средства интеграции неоднородных баз данных [Текст] / Калиниченко Л.А. – Москва: Наука, 1983. – 420 с.

87. Калиниченко Л.А. СИНТЕЗ – язык определения, проектирования и программирования интероперабельных среднеоднородных информационных ресурсов [Текст] / Калиниченко Л.А. – М.: ИПИ РАН, 1993. – 115 с.

88. Калиниченко Л.А. Методы синтеза канонических моделей, предназначенных для достижения семантической интероперабельности неоднородных источников информации [Текст] / Калиниченко Л.А., Ступников С.А., Земцов Н.А. – Москва: ИПИ РАН, 2005. – 84 с.

89. Карпенко А.С. Логика на пороге нового тысячелетия [Електронний ресурс] / Карпенко А.С. – Режим доступу: http://kiev.philosophy.ru/library /logic/karpenko/01.html.

90. Катренко А.В. Методи видобування знань в інформаційних системах [Текст] / Катренко А.В. // Вісник Державного університету «Львівська політехніка». – Л.: Вид-во Державного ун-ту «Львівська політехніка», 1999. – № 383: Інформаційні системи та мережі. – С. 96–101.

91. Катренко С.А. Застосування формальних моделей та методів у видобуванні та виявленні знань зі сховищ даних [Текст] / Катренко С.А., Буров Є.В. // Вісник Національного університету «Львівська політехніка». – Л.: Вид-во Національного ун-ту «Львів. політехніка», 2000. – № 406: Інформаційні системи та мережі. – С. 156–164.

92. Ким Г. Формат электронных документов EDI ANSI ASC X12 [Електронний ресурс] / Ким Г. – Режим доступу: http://www.citforum.ru/consulting/docflow/edi/.

93. Кицмей Т.В. Представлення нечіткої інформації в базах даних реляційного типу: автореф. дис. канд. техн. наук: 10.05.93 [Текст] / Кицмей Т.В.; Державний інститут «Львівська політехніка». – Львів, 1993. – 22 с.

94. Кравець Р.Б. Застосування багатозначної логіки для інтелектуального аналізу даних [Текст] / Кравець Р.Б., Шаховська Н.Б. // Вісник Національного університету «Львівська політехніка». – Л.: Вид-во Національного ун-ту «Львівська політехніка», 2002. – №468: Комп'ютерна інженерія та інформаційні технології. – С. 58–65.

95. Кравець Р.Б. Система вибору оптимальних даних до регулювання перетоку електроенергії на експорт [Текст] / Кравець Р.Б., Шаховська Н.Б., Продан А. // Вісник Національного університету «Львівська

політехніка». – Л.: Вид-во Національного ун-ту «Львівська політехніка», 2004. – № 519: Інформаційні системи та мережі. – С. 206–214.

96. Крайовський В.Я. Використання онтологічного підходу та простору даних для автоматизованого реферування текстових документів [Текст] / В.Я.Крайовський, В.В.Литвин, Н.Б.Шаховська. // Системний аналіз та інформаційні технології: матеріали Міжнародної науково-технічної конференції (SAIT 2009), 23-28 травня 2009 р., Київ / ННК ІПСА НТУУ КПІ. – К.: НТУУ КПІ, 2009. – С. 125.
97. Крайовський В.Я. Основні підходи до розроблення програмного комплексу автоматичного реферування текстів [Текст] / Крайовський В.Я., Литвин В.В., Шаховська Н.Б. // Збірник наукових праць Інституту проблем моделювання в енергетиці / Національна академія наук України, Відділення фізико-технічних проблем енергетики, Інститут проблем моделювання в енергетиці. – Київ, 2009. – Вип.51. – С. 178–186.
98. Кузнецов С. Пространства данных: исследовательский полигон или путь к новому поколению систем управления данными? [Електронний ресурс] / Кузнецов С. – Режим доступу: http://synthesis.ipi.ac.ru/sigmod/seminar/s20060420.
99. Кулик Б. А. Вероятностная логика на основе алгебры кортежей [Текст] / Кулик Б. А. // Известия РАН. Теория и системы управления. – 2007. – № 1. – С. 118–127.
100. Кулик Б.А. Представление логических систем в вероятностном пространстве на основе алгебры кортежей. 1. Основы алгебры кортежей [Текст] / Кулик Б. А. // Автоматика и телемеханика. – 1997. – № 1. – С. 126–136.
101. Кулик Б.А. Представление логических систем в вероятностном пространстве на основе алгебры кортежей. 2. Измеримые логические системы [Текст] / Кулик Б.А., Наумов М. В. // Автоматика и телемеханика. – 1997. – № 2. – С. 169–179.

102. Лагозе К. Инициатива «Открытые архивы»: создание среды с высокой степенью интероперабельности [Електронний ресурс] / Лагозе К. // Электронные библиотеки. – 2001. – Т. 4. – Вып. 6. – Режим доступу: http://www.elbib.ru/index.phtml?page=elbib/rus/journal/2001/part6/LS.

103. Левин Н. А. Алгебра многомерных матриц как универсальное средство моделирования данных и ее реализация в современных СУБД [Текст] / Левин Н. А., Мунерман В. И., Сергеев В. П. // Системы и средства информатики. – Москва: Наука, 2004. – Вып. 14. – С. 86–99.

104. Липаев В. Анализ качества баз данных [Текст] / Липаев В. // Открытые системы. – 2002. – №3 – С. 54.

105. Липинсон С. Cloud Computing: информация и процессы [Електронний ресурс] / Липинсон С. // Открытые системы. – 2008. – № 11. – Режим доступу: http://www.osp.ru/os/2008/08/1858076073/_p2.html.

106. Литвак Б.Г. Разработка управленческих решений [Текст]/Литвак Б.Г. – М.: Дело, 2000. – 392 с.

107. Литвак Б.Г. Экспертная информация: Методы получения и анализа [Текст] / Литвак Б.Г. – М.: Радио и связь, 1982. –184 с.

108. Литвин В.В. Про задачу автоматизованого анотування події на основі простору даних [Текст] / Литвин В.В., Шаховська Н.Б. // Науковий вісник Чернівецького університету. Серія Фізика. Електроніка: збірник наукових праць. – Чернівці, 2008. – № 426: Тематичний випуск «Комп'ютерні системи та компоненти». – С. 58–63.

109. Проектування інтелектуальних агентів на основі адаптивних онтологій / В.В.Литвин, Н.Б.Шаховська, А.С.Мельник, О.Ю. Пшеничний та ін. [Текст] // Матеріали міжнародної наукової конференції «Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту» (ISDMCI'2010, 17–21 травня 2010 р.), Євпаторія. – Т.2. – Херсон, 2010. – С. 401–404.

110. Литвин В.В. Реферування текстових документів на основі зважування міри TF-IDF онтологією предметної галузі [Текст] / Литвин В.В., Шаховська Н.Б., Крайовський В.Я. // Вісник Національного університету «Львівська політехніка». - Л. : Вид-во Нац. ун-ту

«Львівська політехніка», 2010. – №689: Інформаційні системи та мережі. – С. 154–164.

111. Лондон Дж. Управление информационными системами 7-е издание [Текст] / Дж. Лондон, К.Лондон. – СПб.:Питер, 2005. – 912 с.

112. Малюта Т. А. Методи і засоби роботи з неповними і нечіткими значеннями в системах реляційних баз даних: автореф. дис. канд. техн. наук: 08.10.93 [Текст] / Малюта Т.А.; Державний інститут «Львівська політехніка». – Львів, 1993. – 24 с.

113. Мальцев А.И. Алгебраические системы [Текст] / Мальцев А.И. – М.: Наука, 1970. – 392 с.

114. Матов О. Я. Сучасні технології інтеграції інформаційних ресурсів [Текст] / О. Я. Матов, І. О. Храмова // Реєстрація, зберігання і обробка даних. – 2009. – Т. 11, № 1. – С. 33–42.

115. Медиковський М.О. Формалізація операцій над джерелами даних у просторі даних [Текст] / М.О. Медиковський, Н.Б. Шаховська // Оптико-електронні інформаційно-енергетичні технології: міжнародний науково-технічний журнал. – Вінниця, 2009. – №1(17). – С. 49–58.

116. Мельникова Н.І. Аналітичний огляд засобів програмного забезпечення в медичній галузі [Текст] / Мельникова Н.І., Шаховська Н.Б. // Вісник Національного університету «Львівська політехніка». – Л. : Вид-во Нац. ун-ту «Львівська політехніка», 2010. – № 673: Інформаційні системи та мережі. – С. 146–154.

117. Методы и модели оценивания качества программного обеспечения [Текст] / Воробьев В. И., Копыльцов А. В., Пальчун Б. П., Юсупов Р. М. – С-Пб.: СПИИРАН. – 1992. – 33 с.

118. Михайлов В. С. Теория управления [Текст] / Михайлов В. С. – Киев: Выща школа, Головное издательство, 1988. – 608 с.

119. Михайлов К. Особливості управління програмою розвитку системи вищої освіти регіону [Текст]/Михайлов К., Михелєв І., Шаховська Н. // Вісник Національного університету «Львівська політехніка». – 2010. – № 672: Комп'ютерні науки та інформаційні технології. – С. 372–378.

120. Моисеев Н.Н. Элементы теории оптимальных систем [Текст] / Н.Н.Моисеев. – М.: Наука.– 1975. – 528 с.

121. Найт Ф.Х. Риск, неопределенность и прибыль [Текст] / Ф.Х.Найт. – М.: Дело, 2003. – 358 с.

122. Научная информация [Електронний ресурс]. – Режим доступу: http://www.elbib.ru/index.phtml?env_page=methodology/metadata/md_review/md_descrip_scientific.html.

123. Некрасов В. Основные концепции и подходы при создании контекстно-поисковых систем на основе реляционных баз данных [Електронний ресурс]. – 2006. – Режим доступу: http://www.citforum.ru/ database/articles/ search_sys.shtml.

124. Новицкий А.В. Пример построения научных архивов с помощью EPrints [Текст] / А.В. Новицкий, В.А. Резниченко, Г.Ю. Проскудина // RCDL'2006, 17–19 Октября, Суздаль, Россия. – Суздаль, 2006. – С. 154−161.

125. Новицкий А.В. Создание научных архивов с помощью системы Eprints [Електронний ресурс] / Новицкий А.В., Резниченко В.А., Проскудина Г.Ю. // Электронные библиотеки. – 2006. – Т.9, вып. 4. – Режим доступу: http://www.elbib.ru/index.phtml?page=elbib/rus/journal/ 2006/ part4/ Novitski.

126. Jalal-Kamali A. How to Understand Connections Based on Big Data / Jalal-Kamali A., Hossain M.S., Kreinovich V. // Cliques to Flexible Granules / Department of Computer Science, University of Texas at El Paso, El Paso. – 2014. – Т.10. – P. 63–87.

127. Brassard J.-P. Path Building in cellular partitioning networks [Text] / Brassard J.-P., Gecsei J. // ACM SIGARCH Computer Archit News. – 1980. – № 8(3). – P. 44–50.

128. Advanced Statistical Methods for the Analysis of Large Data [Text] / Di Ciaccio A., Coli M., Angulo Ibanez J.M. (eds.). – Berlin: Springer, 2012. – 136 p.

129. : Rex: explaining relationships between entity pairs [Text] / Fang L., Sarma A.D., Yu C., Bohannon P. // Proc. VLDB Endowment. – 2011. – Vol.5(3). – P. 241–252.

130. Connecting the dots between PubMed abstracts [Text] / Hossain M.S., Gresock J., Edmonds Y., Helm R., Potts M., Ramakrishnan N. // PLoS ONE. – 2012. – Vol.7(1). – Paper e29509.

131. Service-generated big data and big data-as-a-service: an overview [Text] / Borkar V., Zibin Z., Jieming Z., Lyu M.R. // IEEE International Congress on Big Data (BigData Congress), 27 June–2 July 2013, Santa Clara. – Santa Clara, 2013. – P. 403, 410.

132. The big picture, from grids and clouds to crowds: a data collective computational intelligence case proposal for managing disasters [Text] / Bu Y., Bessis N., et al.: // International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 4-6 Nov. 2010, New York. – New York, 2010. – P.351-357.

133. A demonstration of SciDB: a science-oriented DBMS [Text] / Cudré-Mauroux, P., et al. // Proc. VLDB Endowment. – 2009. – Vol. 2(2). – P. 1534–1537.

134. Чистов С.М. Державне регулювання економіки [Текст] / С.М.Чистов, А.Є. Никифоров, Т.Ф.Куценко. – К.: КНЕУ, 2000. – 516 с.

135. «Big Data» [Електронний ресурс]. Режим доступу: http://www.tadviser.ru/ index.php/Статья:Большие_данные_%28Big_Data%29#cite_note-g-6.

136. Frank A. Ohlhorst Cloudy Year for Big Data. eWeek [Electronic Resours] / Frank A. – Access mode: http://www.eweek.com/c/a/Cloud-Computing/ 2012-A-Cloudy-Year-for-Big-Data-102807.

137. Фаулер М. UML в кратком изложени / М. Фаулер, К. Скотт. – М. : Мир. – 1999. – 340 с.

138. The Open Archives Initiative Protocol for Metadata Harvesting Protocol Version 2.0 of 2002-06-14 eWeek [Electronic Resours]. – Access mode: http://www.openarchives.org/OAI/2.0 /openarchivesprotocol.htm.

139. Big Data от А до Я. Часть 1: Принципы работы с большими данными, парадигма MapReduce [Електронний ресурс]. – Режим доступу: https://habrahabr.ru/company/dca/blog/267361.

140. A Novel Multidimensional Approach to Integrate Big Data in Business Intelligence / Maté, Alejandro, et al. // Journal of Database Management (JDM). – 2015. – Vol. 26.2. – P.14-31.

141. Digital universe of opportunities [Електронний ресурс]. – Режим доступу: http://www.emc.com/leadership/digital-universe/index.htm?pid=landing-digitaluniverse-131212.

Додаток А

Акти впровадження

результатів дисертаційної роботи

ЗАТВЕРДЖУЮ

Директор Золочівського коледжу
Національного університету
«Львівська політехніка»
_____ Я.Є. Жулин
11.09.2015р.

АКТ
про впровадження в навчальний процес результатів
кандидатської дисертаційної роботи
Болюбаша Юрія Ярославовича

Цей акт складено про те, що результати кандидатської дисертаційної роботи Болюбаша Юрія Ярославовича впроваджено у навчальний процес підготовки фахівців за спеціальностями 5.05010301 «Розробка програмного забезпечення» та 5.05010101 «Обслуговування програмних систем і комплексів» Золочівського коледжу Національного університету "Львівська політехніка".

Впровадження результатів кандидатської роботи полягає в їхньому використанні при викладанні навчальних дисциплін як окремих розділів лекційних курсів, так і в циклах лабораторних робіт.

Зокрема для викладання дисципліни «Бази даних» для студентів освітньо-кваліфікаційного рівня «молодший спеціаліст», що навчаються за спеціальністю 5.05010301 "Розробка програмного забезпечення" використано такі результати:

- методологічні основи побудови просторів даних;
- методи інтеграції даних з різних джерел;
- системи зберігання даних;
- поняття про інтелектуальний аналіз даних.

У лекційному курсі «Основи баз даних та знань» для студентів освітньо-кваліфікаційного рівня «молодший спеціаліст», що навчаються за спеціальністю 5.05010101 "Обслуговування програмних систем і комплексів", використано такі результати:

- рівні моделювання даних;
- поняття консолідованої інформації та її відображення в сховищах даних;
- характеристика даних сховища, структура сховища даних.

Заступник директора з
навчально-виховної роботи М.Б. Рудий

Голова циклової комісії
природничо-математичних
та комп'ютерних дисциплін Р.В. Вовнянка

Викладач комп'ютерних дисциплін І.П. Чіпак

ЗАТВЕРДЖУЮ
Голова Золочівської райдержадміністрації
_____ В.Й. Недзельський
"26" березня 2015 р.

АКТ

про впровадження результатів дисертаційної роботи
здобувача кафедри інформаційних систем і мереж
Національного університету «Львівська політехніка»
Болюбаша Юрія Ярославовича

Цей акт складений про те, що результати кандидатської дисертаційної роботи Болюбаша Ю.Я. використані при розробленні прототипу комплексної інформаційної системи, призначеної для збору, аналізу та планування базових показників економічного розвитку Золочівщини.

Впровадження результатів дисертаційної роботи Болюбаша Ю.Я. полягає у розробленні:

- методу інтеграції соціо-еколого-економічних даних за рахунок визначення пари «сутність-характеристика» та узгодження семантики, що на відміну від методів інтеграції даних на рівні сховища даних дозволило інтегрувати дані з джерел з наперед невідомою структурою даних без попереднього локального завантаження, і що, в свою чергу, дало змогу підвищити ефективність подальшого аналізу Великих даних;
- розроблено механізми підтримки прийняття рішень щодо керування розвитком інфраструктури, які дозволяють визначати слабкі ланки у діяльності регіону та на основі цього пропонують методи вирішення знайдених проблем.

Зазначені результати дисертаційних досліджень, що втілені в

розробленому прототипі системи, дозволили:

- підвищити оперативність отримання, аналізу та використання інформації, необхідної для підтримки прийняття рішень щодо керування регіоном;
- сформувати єдину систему назв та позначень об'єктів регіону, що дозволило уніфікувати та ефективно опрацьовувати інформацію від них;
- підвищити якість та дієвість керуючих рішень в регіоні за рахунок оперування достовірною інформацією, отриманою безпосередньо від джерела;
- своєчасно виявляти та відхиляти негативні тенденції розвитку.

Члени комісії

Голова Золочівської районної ради А.В. Леськів

Начальник відділу економічного розвитку
і торгівлі Золочівської райдержадміністрації О.П. Залуцька

Начальник відділу розвитку
інфраструктури та будівництва
Золочівської райдержадміністрації О.Я. Перейма

РЕГІОНАЛЬНА АГЛОМЕРАЦІЯ
«ДРОГОБИЧЧИНА»

82100, м. Дрогобич, пл. Ринок, 1, тел. /факс (03244) 45-12-17
e-mail:droup@mail.ru

№ 37

від «15» квітня 20 16 р.

Довідка

про впровадження результатів наукового дослідження

Болюбаша Юрія Ярославовича

Науково-технологічна рада Регіональної агломерації «Дрогобиччина» під час розроблення пріоритетних напрямів та програми розвитку Дрогобиччини, для комплексного аналізу регіону та прогнозування його економічного, соціального та екологічного розвитку, використала результати дисертаційного дослідження Болюбаша Юрія Ярославовича на тему «Методи та засоби опрацювання Великих даних в системах територіального управління», зокрема:

1. інформаційну модель Великих даних “сутність-характеристика”;
2. метод перетворення реляційних та слабоструктурованих даних в дані, подані в моделі “сутність-характеристика”;
3. програмні компоненти системи підтримки прийняття рішень для управління регіоном з використанням Великих даних.

Результати дослідження підтвердили дієвість керуючих рішень в регіоні за рахунок оперування достовірною інформацією, отриманою безпосередньо від джерела; результативність алгоритму отримання даних з різних джерел та їх аналізу; ефективність засобів для розрахунку та аналізу основних показників розвитку регіону.

Таким чином, можна дійти висновку, що результати дисертаційного дослідження Болюбаша Юрія Ярославовича можуть знайти широке застосування для розроблення програмних компонент процесів управління регіонами України та України загалом.

Директор “Регіональної агломерації Дрогобиччина” І.П. Головкевич

«Затверджую»
Проректор з наукової роботи
Національного університету
«Львівська політехніка»
_____ проф. Н.І. Чухрай
29. 04. 2016 р.

А К Т

про використання результатів дисертаційної роботи ***«Методи та засоби опрацювання інформаційних ресурсів Великих даних в системах територіального управління»*** здобувача кафедри інформаційних систем та мереж Болюбаша Юрія Ярославовича, представленої на здобуття наукового ступеня кандидата технічних наук, при виконанні науково-дослідної роботи Національного університету «Львівська політехніка»

Ми, що нижче підписалися, начальник НДЧ, к.т.н., доц. Жук Л.В. та члени комісії: завідувач відділу науково-організаційного супроводу наукових досліджень, к.т.н. Лазько Г.В., заступник начальника планово-фінансового відділу Чулой Т.М. та завідувач кафедри інформаційних систем та мереж, д.т.н., проф. Литвин В.В. цим актом підтверджуємо, що результати дисертаційного дослідження здобувача кафедри інформаційних систем та мереж Болюбаша Ю.Я. використано під час виконання науково-дослідної роботи Національного університету "Львівська політехніка", зокрема:

- «Комплекс інтелектуальних інформаційних технологій інтеграції даних для обліку та аналізу підвищення кваліфікації вчителів», номер держреєстрації 0113U005273

В рамках науково-дослідної роботи Болюбаш Ю.Я. розробив модель даних «сутність-характеристика» для подання та аналізу різнорідних даних; розробив методи перетворення реляційних та слабоструктурованих даних у дані, подані в моделі «сутність-характеристика», що дало змогу уніфікувати форму запитів до різних моделей даних; розробив федеративний метод інтеграції даних через визначення пари «сутність-характеристика», який став основою для розроблення методу формування відповіді на запит користувача; розробив метод формування відповіді на запит користувача до Великих даних шляхом уніфікації елементів мов запитів до різних інформаційних джерел, що дало змогу трансформувати запит до Великих даних у запит на основі ключових слів; розробив програмні компоненти інформаційної системи підтримки прийняття рішень «Інтегратор» для управління регіоном з використанням Великих даних.

Використання розроблених Болюбаш Ю.Я. методів та засобів опрацювання та аналізу різнорідної інформації на основі Великих даних дало змогу удосконалити алгоритми інтеграції інформаційних ресурсів за допомогою попереднього визначення

структури джерел даних та їх узгодження, що дало можливість розробити алгоритм оцінювання якості Великих даних. Зокрема, алгоритм пошуку даних за запитом користувача до різнотипних джерел даних, дозволив збільшити релевантність відповіді користувачеві на 2%. Спроектована архітектура інформаційної системи для роботи з Великими даними, дає можливість реалізувати окремі компоненти та функціональні модулі для аналізу даних та прогнозування показників розвитку регіону.

Начальник НДЧ,
канд. техн. наук, доц. Л.В. Жук

Члени комісії:
зав. відділу НОСНД,
канд. техн. наук Г.В. Лазько

Заст. нач. ПФВ Т.М. Чулой

Зав. кафедри
інформаційних систем та мереж,
д. т. н., проф. В.В. Литвин

Додаток Б

Програмний код

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Text.RegularExpressions;
using Word = Microsoft.Office.Interop.Word;

namespace Wp1
{
    public partial class Form1 : Form
    {

//=============================================================================================
        int ind = 54;
        int nos = 5;
        string[] KWColumns = new string[] { "WordID", "Word", "Frequency", "Place",
"Format", "UserWeight", "Sum" };
        string[] SenColumns = new string[] { "SentenceID", "WordsWeight", "Format", "Place",
"Sum" };
        string[] WSenColumns = new string[] { "ID", "WordID", "SentenceID" };

//=============================================================================================
        const string c_vower = "аеиіоуыэюяї";
        const string c_perfectiveground =
"((ив|ивши|ившись|ыв|ывши|ывшись)|((?<=[ая])(в|вши|вшись)))$";
        const string c_reflexive = "(с[яь])$";
        const string c_adjective =
"(ее|ие|ые|ое|ими|ыми|ей|ий|ый|ой|ем|им|ым|ом|его|ого|еых|ую|юю|ая|яя|ою|ею|ому|ій|ім|е|є|а|
ої|у|я|ю|их|іх|іми|их|ох|ома)$";
        const string c_participle = "((ивш|ывш|ующ)|((?<=[ая])(ем|нн|вш|ющ|щ)))$";
        const string c_verb =
"((ила|ыла|ена|ейте|уйте|ите|или|ыли|ей|уй|ил|ыл|им|ым|ены|ить|ыть|ишь|ую|ю|ать|ять|еш|е|у|є
ш|є|иш|емо|уть|емо|єте|ють|имо|ите|уть)|((?<=[ая])(ла|на|ете|йте|ли|й|л|ем|н|ло|но|ет|ют|ны|
ть|ешь|нно)))$";
        const string c_noun =
"(а|ев|ов|ие|ье|е|иями|ями|ами|еи|ии|и|ией|ей|ой|ий|й|и|иям|ям|ием|ем|ам|ом|о|у|ах|иях|ях|ы|
ь|ию|ью|ю|ия|ья|я|і|ою|ею|ень|ї|ею|еві|ові|ів|яти|яті|ені|ята|ятам|ятами)$";
        const string c_rvre = "^(.*?[аеиоуыэюя])(.*)$";
        const string c_derivational =
"[^аеиоуыэюя][аеиоуыэюя]+[^аеиоуыэюя]+[аеиоуыэюя].*(?<=о)сть?$";

//=============================================================================================
        void deleteval(string tabname)
        {
            SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-pc;Initial
Catalog=db1;Integrated Security=True");
            SqlCommand cmd = new SqlCommand();
            SqlDataReader reader;
            cmd.CommandText = "delete from "+tabname+";";
            cmd.CommandType = CommandType.Text;
```

```
            cmd.Connection = sqlConnection1;

            sqlConnection1.Open();

            reader = cmd.ExecuteReader();

            sqlConnection1.Close();
        }

//=============================================================================================
        public Form1()
        {
            InitializeComponent();
            deleteval("Words_Sentence");
            deleteval("Keywords");
            deleteval("Sentence");
        }

//=============================================================================================
        int KWNum(string tabname,string idname)
        {
            SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-pc;Initial
Catalog=db1;Integrated Security=True");
            SqlCommand cmd = new SqlCommand();
            Object returnValue;
            cmd.CommandText = "select count(*) from "+tabname+" where
"+tabname+"."+idname+">0;";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = sqlConnection1;

            sqlConnection1.Open();

            returnValue = cmd.ExecuteScalar();

            sqlConnection1.Close();
            return int.Parse(returnValue.ToString());
        }

//=============================================================================================

        private void Form1_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the 'sWDataSet.StopWords' table. You
can move, or remove it, as needed.
            this.stopWordsTableAdapter.Fill(this.sWDataSet.StopWords);
            // TODO: This line of code loads data into the 'db1DataSet.Words_Sentence'
table. You can move, or remove it, as needed.
            this.words_SentenceTableAdapter.Fill(this.db1DataSet.Words_Sentence);
            // TODO: This line of code loads data into the 'db1DataSet.Sentence' table. You
can move, or remove it, as needed.
            this.sentenceTableAdapter.Fill(this.db1DataSet.Sentence);
            // TODO: This line of code loads data into the 'db1DataSet.Keywords' table. You
can move, or remove it, as needed.
            this.keywordsTableAdapter.Fill(this.db1DataSet.Keywords);

        }

//=============================================================================================
         void SWInsertWord(string value)//for table stopwords
        {
            ind=ind+1;
            SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-pc;Initial
Catalog=SW;Integrated Security=True");
            SqlCommand cmd = new SqlCommand();
```

```
            SqlDataReader reader;
            cmd.CommandText = "insert into StopWords(ID, Word) values ("+ ind +", '" + value
+ "')";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = sqlConnection1;

            sqlConnection1.Open();

            reader = cmd.ExecuteReader();

            sqlConnection1.Close();
        }

//============================================================================================
        void DBInsertWord(string value)//for table keywords
        {
            ind=KWNum("Keywords","WordID")+1;
            SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-pc;Initial
Catalog=db1;Integrated Security=True");
            SqlCommand cmd = new SqlCommand();
            SqlDataReader reader;
            cmd.CommandText = "insert into Keywords(WordID, Word) values ("+ ind +", '" +
value + "')";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = sqlConnection1;

            sqlConnection1.Open();

            reader = cmd.ExecuteReader();

            sqlConnection1.Close();
        }

//============================================================================================
        void WSInsertWord(int wordid,int senid)//for table Words_Sentence
        {
            ind = KWNum("Words_Sentence", "ID") + 1;
            SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-pc;Initial
Catalog=db1;Integrated Security=True");
            SqlCommand cmd = new SqlCommand();
            SqlDataReader reader;
            cmd.CommandText = "insert into Words_Sentence(ID, WordID, SentenceID) values ("
+ ind + ", " + wordid + ", " + senid + ")";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = sqlConnection1;

            sqlConnection1.Open();

            reader = cmd.ExecuteReader();

            sqlConnection1.Close();
        }

//============================================================================================
        void DBCreateSEN(int value)//for table sentence
        {
            int ind=0;
            for (int i = 0; i < value; i++)
            {
                ind = i + 1;
                SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-
pc;Initial Catalog=db1;Integrated Security=True");
                SqlCommand cmd = new SqlCommand();
                SqlDataReader reader;
```

```
                cmd.CommandText = "insert into Sentence(SentenceID) values (" + ind + ")";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = sqlConnection1;

                sqlConnection1.Open();

                reader = cmd.ExecuteReader();

                sqlConnection1.Close();
            }
        }

//=============================================================================================
        void DBUpdateKWValue(string value,string colname,string newval)//for table keywords
        {
            newval=newval.Replace(',','.');
            SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-pc;Initial
Catalog=db1;Integrated Security=True");
            SqlCommand cmd = new SqlCommand();
            SqlDataReader reader;
            cmd.CommandText = "update Keywords set "+colname+"="+newval+" where
Keywords.Word='"+value+"';";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = sqlConnection1;

            sqlConnection1.Open();

            reader = cmd.ExecuteReader();

            sqlConnection1.Close();
        }

//=============================================================================================
        void DBUpdateSenValue(int value, string colname, string newval)//for table sentence
        {
            newval = newval.Replace(',', '.');
            SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-pc;Initial
Catalog=db1;Integrated Security=True");
            SqlCommand cmd = new SqlCommand();
            SqlDataReader reader;
            cmd.CommandText = "update Sentence set " + colname + "=" + newval + " where
Sentence.SentenceID=" + value + ";";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = sqlConnection1;

            sqlConnection1.Open();

            reader = cmd.ExecuteReader();

            sqlConnection1.Close();
        }

//=============================================================================================
        void DBSetWWSen(int value)//for table sentence
        {
            SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-pc;Initial
Catalog=db1;Integrated Security=True");
            SqlCommand cmd = new SqlCommand();
            SqlDataReader reader;
            cmd.CommandText = "update Sentence set WordsWeight =(select SUM(sum) from
(select Keywords.Sum from Keywords, Words_Sentence where
Keywords.WordID=Words_Sentence.WordID and Words_Sentence.SentenceID=" + value + ") as
k)where Sentence.SentenceID=" + value + ";";
            cmd.CommandType = CommandType.Text;
```

```
            cmd.Connection = sqlConnection1;

            sqlConnection1.Open();

            reader = cmd.ExecuteReader();

            sqlConnection1.Close();
        }

//=============================================================================================
        string BDGetKWValue(string value, string colname)//for table keywords
        {
            SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-pc;Initial
Catalog=db1;Integrated Security=True");
            SqlCommand cmd = new SqlCommand();
            Object returnValue;
            cmd.CommandText = "select Keywords."+colname+" from Keywords where
Keywords.Word='" + value + "';";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = sqlConnection1;

            sqlConnection1.Open();

            returnValue = cmd.ExecuteScalar();

            sqlConnection1.Close();
            return returnValue.ToString();
        }

//=============================================================================================
        int BDGetKstKW(int value)//get number of keywords in sentence
        {
            SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-pc;Initial
Catalog=db1;Integrated Security=True");
            SqlCommand cmd = new SqlCommand();
            Object returnValue;
            cmd.CommandText = "select COUNT(Words_Sentence.WordID) from Words_Sentence where
Words_Sentence.SentenceID="+value.ToString()+";";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = sqlConnection1;

            sqlConnection1.Open();

            returnValue = cmd.ExecuteScalar();

            sqlConnection1.Close();
            return int.Parse(returnValue.ToString());
        }

//=============================================================================================
        string BDGetKWWord(int value, string colname)//for table keywords
        {
            SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-pc;Initial
Catalog=db1;Integrated Security=True");
            SqlCommand cmd = new SqlCommand();
            Object returnValue;
            cmd.CommandText = "select Keywords." + colname + " from Keywords where
Keywords.WordID=" + value + ";";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = sqlConnection1;

            sqlConnection1.Open();

            returnValue = cmd.ExecuteScalar();
```

```
            sqlConnection1.Close();
            return returnValue.ToString();
        }

//=============================================================================================
===         string BDGetSenValue(int value, string colname)//for table sentence
        {
            SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-pc;Initial
Catalog=db1;Integrated Security=True");
            SqlCommand cmd = new SqlCommand();
            Object returnValue;
            cmd.CommandText = "select Sentence." + colname + " from Sentence where
Sentence.SentenceID=" + value + ";";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = sqlConnection1;

            sqlConnection1.Open();

            returnValue = cmd.ExecuteScalar();

            sqlConnection1.Close();
            return returnValue.ToString();
        }

//=============================================================================================
        int BDBoolKWValue(string value)
        {
            SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-pc;Initial
Catalog=db1;Integrated Security=True");
            SqlCommand cmd = new SqlCommand();
            Object returnValue;
            cmd.CommandText = "select count(*) from Keywords where Keywords.Word='" + value
+ "';";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = sqlConnection1;

            sqlConnection1.Open();

            returnValue = cmd.ExecuteScalar();

            sqlConnection1.Close();
            return int.Parse(returnValue.ToString());
        }
//=============================================================================================
===         int BDBoolSWValue(string value)
        {
            SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-pc;Initial
Catalog=SW;Integrated Security=True");
            SqlCommand cmd = new SqlCommand();
            Object returnValue;
            cmd.CommandText = "select count(*) from StopWords where StopWords.Word='" +
value + "';";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = sqlConnection1;

            sqlConnection1.Open();

            returnValue = cmd.ExecuteScalar();

            sqlConnection1.Close();
            return int.Parse(returnValue.ToString());
        }
```

```
//=============================================================================================

        float SumKW(string value)//for table keywords
        {
            float suma = 0;
            for(int i=2;i<6;i++)
            {
                suma = suma + float.Parse(BDGetKWValue(value, KWColumns[i]));
            }
            return suma;
        }
//=============================================================================================
===         float SumSen(int value)//for table sentence
        {
            float suma = 0;
            for (int i = 1; i < 4; i++)
            {
                suma = suma + float.Parse(BDGetSenValue(value, SenColumns[i]));
            }
            return suma;
        }
//=============================================================================================
===         void SetSumKW()
        {
            for (int i = 1; i < KWNum("Keywords", "WordID") + 1; i++)
            {
                SqlConnection sqlConnection1 = new SqlConnection("Data Source=user-
pc;Initial Catalog=db1;Integrated Security=True");
                SqlCommand cmd = new SqlCommand();
                Object returnValue;
                cmd.CommandText = "select Keywords.Word from Keywords where
Keywords.WordID=" + i + ";";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = sqlConnection1;

                sqlConnection1.Open();

                returnValue = cmd.ExecuteScalar();

                sqlConnection1.Close();
                //return returnValue.ToString();
                float sum = SumKW(returnValue.ToString());
                DBUpdateKWValue(returnValue.ToString(), "Sum", sum.ToString());
            }
        }
//=============================================================================================
===         void SetSumSen()
        {
            for (int i = 1; i < KWNum("Sentence", "SentenceID") + 1; i++)
            {
                float sum = SumSen(i);
                DBUpdateSenValue(i, "Sum", sum.ToString());

            }
        }

//=============================================================================================

        bool RegexReplace(ref string Original, string Regx, string Value)
        {
            string original = Original;
            Regex reg = new Regex(Regx);
            Original = reg.Replace(Original, Value);
```

```
            return (Original != original);
        }
//==========================================================================================
===         Match RegexMatch(string Original, string Regx)
        {
            Regex reg = new Regex(Regx);
            return reg.Match(Original);
        }
//==========================================================================================
===         MatchCollection RegexMatches(string Original, string Regx)
        {
            Regex reg = new Regex(Regx, RegexOptions.Multiline);
            return reg.Matches(Original);
        }
//==========================================================================================
===         public string Parse(string Query)
        {
            Regex reg = new Regex(@"[ ,\.\?!=\&\*\+]");
            string[] words = reg.Split(Query);
            IList<string> swords = new List<string>();

            for (int i = 0; i < words.Length; i++)
                if (!string.IsNullOrEmpty(words[i].Trim()))
                    swords.Add(Stem(words[i].Trim()));

            string result = string.Join("%", swords.ToArray());

            return string.Format("%{0}%", result);
        }
//==========================================================================================
===         public string Stem(string Word)
        {
            string word = Word.ToLower().Trim().Replace("ë", "e");
            string value = string.Empty;
            do
            {
                MatchCollection matches = RegexMatches(word, c_rvre);
                if (matches.Count < 1)
                {
                    Match matchEnglishOrDigits = RegexMatch(word, "[a-z0-9]");
                    if (matchEnglishOrDigits.Success)
                        value = word;

                    break;
                }


                string rv = matches[0].Value;

                // war 1
                if (!RegexReplace(ref rv, c_perfectiveground, string.Empty))
                {
                    RegexReplace(ref rv, c_reflexive, string.Empty);

                    if (RegexReplace(ref rv, c_adjective, string.Empty))
                        RegexReplace(ref rv, c_participle, string.Empty);
                    else
                        if (!RegexReplace(ref rv, c_verb, string.Empty))
                            RegexReplace(ref rv, c_noun, string.Empty);
                }

                // war 2
                RegexReplace(ref rv, "и$", string.Empty);
```

```
                // шаг 3
                Match match = RegexMatch(rv, c_derivational);
                if (match.Success)
                    RegexReplace(ref rv, "ость?$", string.Empty);

                // шаг 4
                if (!RegexReplace(ref rv, "ь$", string.Empty))
                {
                    RegexReplace(ref rv, "ейше?", string.Empty);
                    RegexReplace(ref rv, "нн$", "н");
                }

                value = rv;

            } while (false);

            return value;
        }


//=============================================================================================
        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                OpenFileDialog path = new OpenFileDialog();
                //making an object for the OpenFileDialog
                path.Title = "Відкрити файл";
                path.InitialDirectory = @"c:\";
                path.RestoreDirectory = true;

                if (path.ShowDialog() == DialogResult.OK)
                {
                    textBox1.Text = path.FileName.ToString();
                }
            }
            catch (Exception l)
            {
                MessageBox.Show(l.Message);
            }
        }

//=============================================================================================
===         private void button2_Click(object sender, EventArgs e)
        {
            deleteval("Words_Sentence");
            deleteval("Keywords");
            deleteval("Sentence");
             try
            {
                Microsoft.Office.Interop.Word.Application wordObject = new
Word.Application();
                object file = textBox1.Text; //this is the path
                object nullobject = System.Reflection.Missing.Value;
                Microsoft.Office.Interop.Word.Document docs = wordObject.Documents.Open
                    (ref file, ref nullobject, ref nullobject, ref nullobject,
                    ref nullobject, ref nullobject, ref nullobject, ref nullobject,
                    ref nullobject, ref nullobject, ref nullobject, ref nullobject,
                    ref nullobject, ref nullobject, ref nullobject, ref nullobject
                                    );
                docs.ActiveWindow.Selection.WholeStory();
                int SCount = docs.ActiveWindow.Selection.Sentences.Count;
                int WCount = docs.ActiveWindow.Selection.Words.Count;
                string word;
```

```
                int ks = 0, form = 0, freq=0,alig=0;
                double frequency = 0,sto=100;
                double place = 0;
                DBCreateSEN(SCount);//creation sentence table
                int i1 = 0, i2 = 0;
                for (int i = 1; i < SCount + 1; i++)
                {
                    if ((docs.ActiveWindow.Selection.Sentences[i].Text.Contains("Ключові
слова:"))||(docs.ActiveWindow.Selection.Sentences[i].Text.Contains("Ключевые слова:"))) ks =
5;
                    else ks = 0;
                    alig=0;
                    if
(docs.ActiveWindow.Selection.Sentences[i].ParagraphFormat.Alignment.ToString().Contains("wdA
lignParagraphCenter")) alig=2;
                    if
(docs.ActiveWindow.Selection.Sentences[i].ParagraphFormat.Alignment.ToString().Contains("wdA
lignParagraphJustify")) alig = 1;
                    if
(docs.ActiveWindow.Selection.Sentences[i].ParagraphFormat.Alignment.ToString().Contains("wdA
lignParagraphLeft")) alig = 0;
                    if
(docs.ActiveWindow.Selection.Sentences[i].ParagraphFormat.Alignment.ToString().Contains("wdA
lignParagraphRight")) alig = 0;
                    DBUpdateSenValue(i, "Format", alig.ToString());
                    alig=0;
                    place = 100 * i / double.Parse(SCount.ToString());
                    if ((place < 10) || (place > 90)) alig = 0;
                    else
                        if ((place < 30) || (place > 70)) alig = 1;
                        else
                            if ((place < 40) || (place > 60)) alig = 2;
                            else
                                alig = 2;
                    DBUpdateSenValue(i, "Place", alig.ToString());
                    if (docs.ActiveWindow.Selection.Text.Contains("Вступ"))
                    {
                        if
(docs.ActiveWindow.Selection.Sentences[i].ParagraphFormat.Alignment.ToString().Contains("wdA
lignParagraphCenter"))
                            i1 = 1;
                        if (docs.ActiveWindow.Selection.Sentences[i].Text.Contains("Вступ"))
                            i2 = 1;
                        if ((i1 == 1) && (i2 != 1) &&
(!docs.ActiveWindow.Selection.Sentences[i].Text.Contains("(c)")) &&
(!docs.ActiveWindow.Selection.Sentences[i].Text.Contains("(C)")) &&
(!docs.ActiveWindow.Selection.Sentences[i].Text.Contains("Ключові слова:")))
                        {
                            alig = 4;
                            DBUpdateSenValue(i, "Place", alig.ToString());
                        }
                    }
                    for (int j = 1; j < docs.ActiveWindow.Selection.Sentences[i].Words.Count
+ 1; j++)
                    {
                        if ((docs.ActiveWindow.Selection.Sentences[i].Words[j].Text.Length >
4) && (BDBoolSWValue(docs.ActiveWindow.Selection.Sentences[i].Words[j].Text) == 0))
                        {
                            word =
Parse(docs.ActiveWindow.Selection.Sentences[i].Words[j].Text);
                            if (BDBoolKWValue(word) == 0)//if its new word
                            {
                                DBInsertWord(word);
                                DBUpdateKWValue(word, "Frequency", "1");
```

```
                                if ((ks > 0) && (j != 1) && (j != 2))
                                    DBUpdateKWValue(word, "Place", ks.ToString());
                                else
                                    DBUpdateKWValue(word, "Place", "0");
                                form = 0;
                    if (docs.ActiveWindow.Selection.Sentences[i].Words[j].Bold == -1) form++;
                    if (docs.ActiveWindow.Selection.Sentences[i].Words[j].Italic == -1) form++;
                    if
(!docs.ActiveWindow.Selection.Sentences[i].Words[j].Underline.ToString().Contains("wdUnderli
neNone")) form++;
                    if
(docs.ActiveWindow.Selection.Sentences[i].ParagraphFormat.Alignment.ToString().Contains("wdA
lignParagraphCenter")) form++;
                                DBUpdateKWValue(word, "Format", form.ToString());
                                DBUpdateKWValue(word, "UserWeight", "0");
                            }
                            else//if word alrady exist
                            {
                                freq = int.Parse(BDGetKWValue(word, "Frequency"));
                                freq++;
                                DBUpdateKWValue(word, "Frequency", freq.ToString());
                                if ((ks > 0) && (j != 1) && (j != 2))
                                    DBUpdateKWValue(word, "Place", ks.ToString());
                                form = 0;
                    if (docs.ActiveWindow.Selection.Sentences[i].Words[j].Bold == -1) form++;
                    if (docs.ActiveWindow.Selection.Sentences[i].Words[j].Italic == -1) form++;
                    if
(!docs.ActiveWindow.Selection.Sentences[i].Words[j].Underline.ToString().Contains("wdUnderli
neNone")) form++;
                    if
(docs.ActiveWindow.Selection.Sentences[i].ParagraphFormat.Alignment.ToString().Contains("wdA
lignParagraphCenter")) form++;
                                freq = int.Parse(BDGetKWValue(word, "Format"));
                    if (freq < form) DBUpdateKWValue(word, "Format", form.ToString());
                            }
                            WSInsertWord(int.Parse(BDGetKWValue(word,"WordID")), i);
                        }
                    }
                }
                for (int i = 1; i < KWNum("Keywords", "WordID")+1; i++)
                {
                    freq = int.Parse(BDGetKWWord(i,"Frequency"));
                    frequency = sto*freq / double.Parse(WCount.ToString());
                 DBUpdateKWValue(BDGetKWWord(i, "Word"), "Frequency", frequency.ToString());
                }
                SetSumKW();
                float WW;
                for (int i = 1; i < KWNum("Sentence", "SentenceID") + 1; i++)
                {
                    DBSetWWSen(i);
                    WW = float.Parse(BDGetSenValue(i, "WordsWeight"));
                    WW = WW / BDGetKstKW(i);
                    DBUpdateSenValue(i, "WordsWeight", WW.ToString());
                }
                SetSumSen();
                int[] isen=new int[nos];
                double max=0,lmax=200;
                int maxi=0;
                if (nos > docs.ActiveWindow.Selection.Sentences.Count)
                    nos = docs.ActiveWindow.Selection.Sentences.Count;
                for (int j = 0; j < nos; j++)
                {
                    max = 0;
                    for (int i = 1; i < KWNum("Sentence", "SentenceID") + 1; i++)
```

```
                    {
                        if ((double.Parse(BDGetSenValue(i, "Sum")) > max) &&
(double.Parse(BDGetSenValue(i, "Sum")) < lmax) &&
(!docs.ActiveWindow.Selection.Sentences[i].Text.Contains("Вступ")) &&
(!docs.ActiveWindow.Selection.Sentences[i].Text.Contains("Висновок")))
                        {
                            max = double.Parse(BDGetSenValue(i, "Sum"));
                            maxi = i;
                        }
                    }
                    isen[j] = maxi;
                    lmax = max;
                }
                string sen = "";
                string[] s=new string[2];
                string conc = "";
                for (int i = 1; i < KWNum("Sentence", "SentenceID") + 1; i++)
                {
                    if (isen.Contains(i))
                    {
                        sen = docs.ActiveWindow.Selection.Sentences[i].Text;
                        if (sen.Contains(','))
                        {
                            int ind = sen.IndexOf(',') + 3;
                            while(sen[ind]!=' ')
                                ind++;
                         conc = sen.Substring(sen.IndexOf(',') + 3, ind - 3 - sen.IndexOf(','));
                            if (BDBoolSWValue(conc)!=0)
                            {
                                s[0] = sen.Substring(0, sen.IndexOf(','));
                                s[1] = sen.Substring(sen.IndexOf(',')+1,sen.Length);
                                if (s[1].Contains(','))
                                {
                                    s[2] = sen.Substring(sen.IndexOf(',') + 1, sen.Length);
                                    sen = s[0] + s[2];
                                }
                                else
                                    sen = s[0].Replace(',','.');
                            }
                        }
                        richTextBox1.AppendText(sen);
                    }
                }
                wordObject.Documents.Close(ref nullobject, ref nullobject, ref nullobject);
                wordObject.Application.Quit(ref nullobject, ref nullobject, ref nullobject);
            }
            catch (Exception j)
            {
                MessageBox.Show(j.Message);
            }
        }

//=============================================================================================
===         private void button3_Click(object sender, EventArgs e)
        {

            try
            {
                SaveFileDialog path = new SaveFileDialog();
                //making an object for the SaveFileDialog
                path.Title = "Зберегти у файл";
                path.InitialDirectory = @"c:\";
                path.RestoreDirectory = true;
                path.Filter = "Документ Word|*.doc,*.docx";
```

```
                path.DefaultExt = "doc,docx";

                if (path.ShowDialog() == DialogResult.OK)
                {
              Microsoft.Office.Interop.Word.Application wordObject = new Word.Application();
                    object file = path.FileName; //this is the path
                    object nullobject = System.Reflection.Missing.Value;
                    Microsoft.Office.Interop.Word.Document docs =
wordObject.Documents.Add(ref nullobject, ref nullobject, ref nullobject, ref nullobject);
                    Word.Paragraph opara;
                    opara = docs.Content.Paragraphs.Add(ref nullobject);
                    opara.Range.Text = richTextBox1.Text;
                    opara.Range.InsertParagraphAfter();
                    docs.ActiveWindow.Selection.WholeStory();
                    docs.ActiveWindow.Selection.Font.Size = 12;
                    docs.ActiveWindow.Selection.ParagraphFormat.Alignment =
Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphJustify;
                    docs.ActiveWindow.Selection.Font.Name = " Segoe UI";
                    docs.ActiveWindow.Selection.Sentences[1].Font.Bold = 1;
                    docs.ActiveWindow.Selection.Sentences[1].Font.Size = 14;
                    docs.ActiveWindow.Selection.Sentences[1].ParagraphFormat.Alignment =
Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;
                    for(int i=1;i<docs.ActiveWindow.Selection.Sentences.Count+1;i++)
                        if (!docs.ActiveWindow.Selection.Sentences[i].Text.Contains("."))

docs.ActiveWindow.Selection.Sentences[i].ParagraphFormat.Alignment =
Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;
                    docs.SaveAs2(ref file, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject);
                    docs.Close(ref nullobject, ref nullobject, ref nullobject);
                }
            }
            catch (Exception l)
            {
                MessageBox.Show(l.Message);
            }
        }
//==========================================================================================
===         private void button4_Click(object sender, EventArgs e)
        {
            string word;
            int weight = 0;
            word = Parse(textBox2.Text);
            weight = int.Parse(textBox3.Text);
            if (BDBoolKWValue(word) == 0)//if its new word
            {
                DBInsertWord(word);
                DBUpdateKWValue(word, "Frequency", "0");
                DBUpdateKWValue(word, "Place", "0");
                DBUpdateKWValue(word, "Format", "0");
                DBUpdateKWValue(word, "UserWeight", weight.ToString());
            }
            else//if word alrady exist
            {
                DBUpdateKWValue(word, "UserWeight", weight.ToString());
            }

        }

//==========================================================================================
private void button5_Click(object sender, EventArgs e)
        {
```

```
            nos = int.Parse(textBox5.Text);
        }

//=============================================================================================

        private void вибертьФайлToolStripMenuItem_Click(object sender, EventArgs e)
        {
            button1_Click(sender,e);
        }

//=============================================================================================

        private void зберегтиУФайлToolStripMenuItem_Click(object sender, EventArgs e)
        {
            button3_Click(sender, e);
        }

//=============================================================================================

        private void Close(object sender, EventArgs e)
        {
            Close();
        }

//=============================================================================================

        private void допомогаToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form2.ActiveForm.Activate();
            Form2 OrdersForm = new Form2();
            OrdersForm.Show();
        }

//=============================================================================================

        private void проАвтораToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form3.ActiveForm.Activate();
            Form3 OrdersForm = new Form3();
            OrdersForm.Show();
        }
    }
}
```