УДК 004.65:004.738.5

М. В. Міняйло
Д. О. Середа

ПІДХОДИ ДО ПОБУДОВИ СИСТЕМ ОБРОБКИ СТРУКТУРОВАНИХ ТА НЕСТРУКТУРОВАНИХ ДАНИХ

Вінницький національний технічний університет

Анотація

Розглянуто підходи до побудови систем обробки даних з урахуванням їх типу: структурованих, напівструктурованих та неструктурованих. Проаналізовано особливості зберігання та обробки JSON-відповідей зовнішніх API, а також документів у форматах PDF та DOCX. Наведено порівняльну характеристику інструментів обробки для кожного типу даних та запропоновано уніфіковану конвеєрну архітектуру (pipeline) для обробки різнорідних даних у рамках єдиної системи.

Ключові слова: структуровані дані; неструктуровані дані; JSON; REST API; обробка документів; Apache Tika; конвеєр даних; напівструктуровані дані.

Abstract

Approaches to building data processing systems are considered with regard to data types: structured, semi-structured, and unstructured. The features of storing and processing JSON responses from external APIs, as well as PDF and DOCX documents, are analyzed. A comparative description of processing tools for each data type is provided, and a unified pipeline architecture for heterogeneous data processing within a single system is proposed.

Keywords: structured data; unstructured data; JSON; REST API; document processing; Apache Tika; data pipeline; semi-structured data.

Вступ

Сучасні інформаційні системи оперують даними принципово різної природи. З одного боку, це чітко структуровані записи реляційних баз даних та API-відповіді у форматі JSON з передбачуваною схемою. З іншого – довільні текстові документи у форматах PDF та DOCX, електронна пошта, зображення та відео. За різними оцінками, понад 80% корпоративних даних є неструктурованими або напівструктурованими [1], що ставить перед розробниками нові архітектурні виклики.

Ключова відмінність між типами даних полягає в наявності або відсутності заздалегідь визначеної схеми. Структуровані дані підпорядковуються жорсткій схемі, що дозволяє ефективно виконувати запити та агрегацію. Неструктуровані дані не мають внутрішньої регулярної організації і потребують спеціалізованих методів попередньої обробки для вилучення корисної інформації [2]. Між цими категоріями розташовані напівструктуровані дані – JSON-відповіді REST API, що мають ієрархічну організацію, але не зобов'язані слідувати фіксованій схемі.

Особливо гостро проблема постає при необхідності інтегрувати різні джерела в єдину аналітичну систему: корпоративні документи, API сторонніх сервісів та реляційні бази даних одночасно. Дослідження підходів до побудови систем, здатних ефективно обробляти всі три типи в єдиній архітектурі, є актуальним завданням.

Основна частина

Структуровані дані характеризуються наявністю чіткої схеми та організацією у вигляді таблиць або впорядкованих ієрархій. Класичним прикладом є реляційні бази даних, де кожен запис підпорядковується суворо визначеному набору полів і типів. У контексті взаємодії з зовнішніми API структурованими вважаються відповіді з передбачуваною схемою. Стандарт JSON визначено у документі RFC 8259, опублікованому IETF у грудні 2017 року [3]. Для обробки таких даних застосовують JSONPath-запити, бібліотеки десеріалізації та ORM-фреймворки.

Напівструктуровані дані займають проміжне положення: вони мають певну ієрархічну організацію (ключі, теги, вкладені об'єкти), але не дотримуються жорсткої схеми. JSON-відповіді реальних API є саме напівструктурованими: поля можуть бути відсутніми, мати різні типи або вкладені об'єкти довільної глибини. Для їх обробки використовується JSON Schema для валідації та

нормалізації, а також документо-орієнтовані СУБД на кшталт MongoDB, які зберігають дані без наперед заданої схеми [2].

Неструктуровані дані – найчисленніша і найскладніша для автоматизованої обробки категорія. До неї належать текстові документи (PDF, DOCX, RTF), зображення, аудіо- та відеофайли, електронна пошта. Обробка таких даних потребує інструментів вилучення вмісту. Інструментальний набір Apache Tika виявляє та вилучає метадані й текст з понад тисячі різних форматів файлів [4]. Apache Tika забезпечує єдиний програмний інтерфейс, делегуючи обробку спеціалізованим бібліотекам: Apache PDFBox для PDF, Apache POI для документів Microsoft Office. Для відсканованих PDF інтегрується рушій оптичного розпізнавання Tesseract OCR [5].

Таблиця 1 – Порівняльна характеристика типів даних та підходів до їх обробки

Тип даних	Приклади форматів	Технологія зберігання	Інструменти обробки	Особливості
Структуровані	Реляційні БД, CSV, JSON з фіксованою схемою, XML-RPC	RDBMS (PostgreSQL, MySQL), колонкові сховища (ClickHouse)	SQL, ORM, JSONPath, pandas DataFrame, jq	Чітка схема, ефективна агрегація, прості запити
Напівструктуровані	JSON/REST API відповіді, YAML, HTML, JSON з довільними полями	MongoDB, Elasticsearch, CouchDB, DynamoDB	JSONSchema, SAX/DOM, BeautifulSoup, json-schema-validator	Гнучка схема, потребує валідації та нормалізації
Неструктуровані	PDF, DOCX, RTF, електронні листи, зображення, аудіо/відео	Object Storage (S3, MinIO), BLOB, файлова система	Apache Tika, PDFBox, Tesseract OCR, NLP-пайплайни	Потребує вилучення тексту й попередньої обробки

Рисунок 1 ілюструє уніфіковану конвеєрну архітектуру системи обробки різнорідних даних. Вхідні дані надходять із трьох джерел: реляційної бази даних, REST API та сховища документів. Рівень маршрутизації визначає тип даних та спрямовує їх до відповідного спеціалізованого обробника. Виходи всіх обробників нормалізуються до єдиного внутрішнього формату та індексуються у спільне сховище для подальшого аналізу та пошуку.

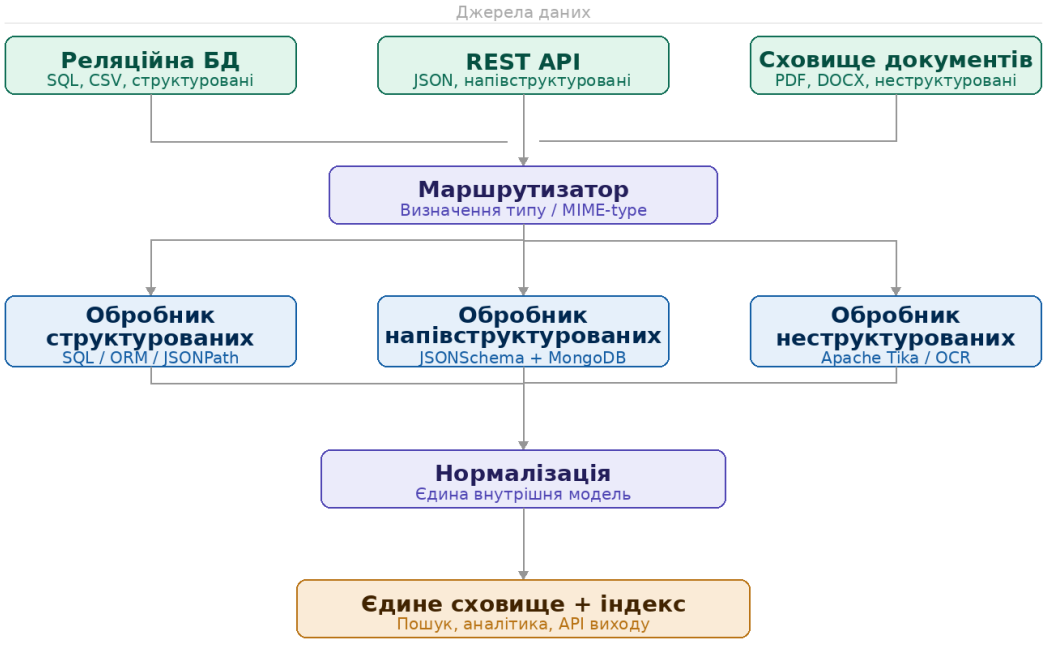

Рисунок 1 – Конвеєрна архітектура системи обробки різнорідних даних

Ключовим архітектурним рішенням є вибір між централізованим та розподіленим підходом. Централізований підхід передбачає обробку всіх типів даних єдиним компонентом; розподілений –

маршрутизацію до спеціалізованих обробників залежно від виявленого типу. Розподілений підхід є кращим для систем із великою кількістю різнорідних джерел, оскільки дозволяє масштабувати кожен обробник незалежно [1, 6].

Для структурованих та напівструктурованих JSON-даних, що надходять через API, типовий конвеєр включає: HTTP-клієнт, десеріалізацію JSON, валідацію за JSON Schema, нормалізацію у внутрішню модель та збереження у СУБД. Для неструктурованих документів конвеєр розширюється стадіями: визначення MIME-типу, вилучення тексту за допомогою Apache Tika, подальша NLP-обробка за потреби та індексування у повнотекстовій пошуковій системі [4, 5].

Важливим аспектом при проектуванні таких систем є обробка нетипових випадків. JSON-відповіді реальних API можуть містити непередбачені поля, тому рекомендується толерантний розбір із логуванням відхилень. Для документів один і той самий формат (наприклад, PDF) може бути текстовим або відсканованим зображенням, що вимагає різних стратегій обробки. Уніфікована архітектура конвеєра з маршрутизацією за типом дозволяє організувати єдиний пошуковий індекс над різнорідними джерелами [1, 6].

Окремої уваги заслуговує питання продуктивності конвеєрної обробки при великих обсягах даних. Для структурованих даних вузьким місцем зазвичай є операції запису в реляційну СУБД. В такому випадку ефективним рішенням є пакетне вставлення (batch insert) замість окремих транзакцій на кожен запис. Для неструктурованих документів найдорожчою операцією є вилучення тексту з PDF, особливо якщо документ містить зображення і потребує OCR. У таких випадках рекомендується розподіляти обробку між кількома процесами або використовувати чергу завдань (task queue) на кшталт Celery чи RabbitMQ, що дозволяє асинхронно обробляти документи без блокування основного потоку [6].

Важливим компонентом системи є шар валідації та контролю якості даних. Для структурованих даних із зовнішніх API доцільно застосовувати схемну валідацію на вході конвеєра: будь-яке відхилення від очікуваної схеми фіксується в журналі помилок і не допускається до подальшої обробки без ручного або автоматичного узгодження. Для напівструктурованих JSON-даних застосовують бібліотеки на кшталт Pydantic (Python) або Joi (Node.js), які дозволяють декларативно описати очікувану структуру об’єкта і автоматично генерувати повідомлення про помилки при невідповідності [2, 3]. Відсутність такого шару на практиці призводить до тихих помилок – некоректні дані потрапляють у сховище і спотворюють результати аналітики.

Зберігання нормалізованих даних у єдиному сховищі відкриває можливості для крос-типового пошуку та аналітики. Наприклад, Elasticsearch дозволяє індексувати як структуровані поля (числові показники, дати, категорії), так і повнотекстовий вміст, вилучений із документів, в одному індексі. Це дає змогу будувати складні запити, що одночасно фільтрують за метаданими та шукають по тексту документів, що є принципово недосяжним при роздільному зберіганні різнотипних даних [1]. Для аналітичних сценаріїв, де важлива агрегація великих обсягів, ефективнішим є колонкове сховище на кшталт ClickHouse або Apache Parquet у зв’язці з Apache Spark.

Спостережуваність (observability) є невід’ємною частиною надійної системи обробки різнорідних даних. Збір метрик часу обробки для кожного типу, відсотка успішно оброблених записів і розміру черги дозволяє своєчасно виявляти деградацію продуктивності та сплески навантаження. Розподілене трасування запитів, реалізоване засобами OpenTelemetry, дає можливість відстежити повний шлях конкретного документа або API-відповіді через усі стадії конвеєра: від вхідного запиту до запису в сховище. Це суттєво спрощує діагностику помилок у складних багатокомпонентних системах [6].

Висновки

Побудова систем обробки різнорідних даних потребує чіткого розмежування між структурованими, напівструктурованими та неструктурованими даними та застосування спеціалізованих інструментів для кожного типу. JSON-дані REST API, незважаючи на зовнішню схожість зі структурованими, на практиці є напівструктурованими і потребують валідації та нормалізації. Документи потребують попередньої стадії вилучення тексту, для якої Apache Tika є зрілим і перевіреним рішенням. Конвеєрна архітектура з маршрутизацією за типом даних дозволяє ефективно інтегрувати обробку різнорідних джерел у рамках єдиної системи та масштабувати кожен її компонент незалежно.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Sakr S. Big Data 2.0 Processing Systems: A Systems Overview. 2nd ed. – Springer, 2020. – 145 p. – ISBN 978-3-030-44186-9. – DOI: 10.1007/978-3-030-44187-6.
2. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. – O'Reilly Media, 2017. – 616 p. – ISBN 978-1-449-37332-0.
3. Bray T. (Ed.) The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259. – Internet Engineering Task Force (IETF), December 2017. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc8259. Дата звернення: 10.03.2026 р.
4. Apache Tika — a content analysis toolkit [Електронний ресурс]. – Apache Software Foundation. – Режим доступу: https://tika.apache.org/. Дата звернення: 12.03.2026 р.
5. Apache Tika — Supported Document Formats [Електронний ресурс]. – Apache Software Foundation. – Режим доступу: https://tika.apache.org/3.1.0/formats.html. Дата звернення: 14.03.2026 р.
6. Adnan K., Akbar R. An analytical study of information extraction from unstructured and multidimensional big data // Journal of Big Data. – 2019. – Vol. 6, No. 1. – P. 1–38. – DOI: 10.1186/s40537-019-0254-8.

Міняйло Михайло Володимирович – студент групи 2ПІ-22б, Факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: mike.min.32@gmail.com.

Середа Дмитро Олександрович – студент групи 2ПІ-22б, Факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: dmytrosereda2004@gmail.com.

Науковий керівник: ***Мельник Олександр Васильович*** – к.т.н., доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, vinncei@gmail.com.

Miniailo Mykhailo Volodymyrovych – student of the group 2PI-22b, Faculty of Information Technology and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: mike.min.32@gmail.com.

Sereda Dmytro Oleksandrovych – student of the group 2PI-22b, Faculty of Information Technology and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: dmytrosereda2004@gmail.com.

Academic supervisor: **Melnyk Oleksandr Vasylovych** – Candidate of Technical Sciences, Associate Professor of the Department of Software, Vinnytsia National Technical University, Vinnytsia, e-mail: vinncei@gmail.com.