УДК 004.62:004.89
DOI https://doi.org/10.32689/maup.it.2025.4.21

Євген ПАТЛАНЬ
аспірант кафедри інформаційних та комп'ютерних систем, Національний університет «Чернігівська політехніка»
ORCID: *0009-0002-4980-4280*

Ірина БІЛОУС
кандидат технічних наук, доцент,
завідувач кафедри інформаційних технологій та програмної інженерії, Національний університет «Чернігівська політехніка»
ORCID: *0000-0003-3092-678X*

МЕТОД АВТОМАТИЗОВАНОЇ МІГРАЦІЇ ДАНИХ МІЖ ВАРІАНТАМИ СХОВИЩ У СИСТЕМАХ З БАГАТОВАРІАНТНОЮ ПЕРСИСТЕНТНІСТЮ

Анотація. Мета дослідження *полягає у розробці концепції та архітектури системи автоматизованої міграції даних, здатної забезпечувати узгодженість, цілісність і еквівалентність інформації під час переходу між різнорідними моделями зберігання в умовах багатоваріантної персистентності. Робота присвячена створенню методу автоматизованої міграції даних у середовищах, де одна й та сама одиниця інформації може існувати у кількох еквівалентних формах.*

Методологія. *Використаний метод базується на формальній моделі факту, що зберігає інваріанти структури, типів і семантики при переході між різними моделями зберігання. Визначено принципи коректних відображень між сховищами та умови, за яких перетворення даних залишаються безвтратними та відтворюваними. Запропоновано систему інваріантів, яка фіксує межі допустимих трансформацій і забезпечує логічну цілісність фактів незалежно від конкретної технології зберігання. Ключовим елементом методу є предметно орієнтована мова опису трансформацій, яка задає правила переходу між моделями даних у формалізованому вигляді. Її конструкції дозволяють виконувати операції проєкції, вкладення, розгортання, з'єднання та індукції зв'язків між структурами; мова має чітку типізацію та забезпечує перевірку коректності на етапах компіляції й виконання. На основі розробленої мови створено прототип системи, який виконує автоматизовану трансформацію даних між реляційними, документними, графовими та key-value сховищами без втручання користувача. Архітектура рішення включає модулі перевірки інваріантів, компіляції трансформацій і виконання міграцій із контролем транзакційності, відкатом і журналом фіксації операцій, а також сервіси оцінки вартості та перевірки коректності.*

Наукова новизна. *Уперше сформовано цілісний формальний метод автоматизованої міграції даних між різновидами сховищ у системах з багатоваріантною персистентністю. Побудовано математичну модель факту, інваріантів і відображень, що визначають межі допустимих перетворень і гарантують семантичну еквівалентність між різними матеріалізаціями даних. Запропоновано предметно орієнтовану мову трансформацій із формальною верифікацією тотальності, типізації та збереження інваріантів; реалізовано архітектуру автоматизованої системи з уніфікованими адаптерами до різнорідних сховищ. У межах експериментів проведено порівняння з ручними трансформаціями та класичним ETL-підходом, що дозволило встановити нульову кількість порушень інваріантів при лише незначному збільшенні часу міграції.*

Висновки. *Запропонований метод характеризується здатністю забезпечувати коректну, відмовостійку та формально перевірену міграцію даних без втрати цілісності. Прототип підтвердив нульовий рівень порушень інваріантів і стабільність результатів у різних сценаріях, а також придатність методу до масштабування. Отримані результати доводять ефективність запропонованого підходу та відкривають перспективи подальшої адаптації системи шляхом використання навчальних агентів та динамічних політик керування міграцією.*

Ключові слова: *інваріанти, десеріалізація, трансформація, відображення, факт.*

Yevhen PATLAN, Iryna BILOUS. AUTOMATED DATA MIGRATION METHOD ACROSS STORAGE VARIANTS IN SYSTEMS WITH MULTIVARIANT PERSISTENCE

Abstract. The purpose *of the research is to develop a concept and architecture of an automated data migration system capable of ensuring consistency, integrity and equivalence of information during the transition between heterogeneous storage models in conditions of multivariate persistence. The work is devoted to the creation of a method of automated data migration in environments where the same unit of information can exist in several equivalent forms.*

Methodology. *The method used is based on a formal model of a fact that preserves invariants of structure, types and semantics during the transition between different storage models. The principles of correct mappings between repositories and the conditions under which data transformations remain lossless and reproducible are determined. A system of invariants is proposed that fixes the boundaries of permissible transformations and ensures the logical integrity of facts regardless of*

the specific storage technology. The key element of the method is a subject-oriented language for describing transformations, which sets the rules for the transition between data models in a formalized form. Its constructions allow performing operations of projection, nesting, expansion, connection and induction of relations between structures; the language has a clear typing and provides correctness checking at the compilation and execution stages. Based on the developed language, a prototype of the system was created, which performs automated data transformation between relational, document, graph and key-value repositories without user intervention. The solution architecture includes modules for checking invariants, compiling transformations and performing migrations with transactional control, rollback and transaction log, as well as cost estimation and correctness checking services.

Scientific novelty. *For the first time, a holistic formal method for automated data migration between types of repositories in systems with multivariate persistence has been formed. A mathematical model of the fact, invariants and mappings has been constructed, which determine the boundaries of permissible transformations and guarantee semantic equivalence between different data materializations. A subject-oriented transformation language with formal verification of totality, typing, and preservation of invariants is proposed; the architecture of an automated system with unified adapters to heterogeneous repositories is implemented. Within the framework of the experiments, a comparison was made with manual transformations and the classical ETL approach, which allowed us to establish a zero number of invariant violations with only a slight increase in migration time.*

Conclusions. *The proposed method is characterized by the ability to provide correct, fault-tolerant, and formally verified data migration without loss of integrity. The prototype confirmed the zero level of invariant violations and the stability of results in various scenarios, as well as the method's suitability for scaling. The results obtained prove the effectiveness of the proposed approach and open up prospects for further adaptation of the system through the use of training agents and dynamic migration management policies.*

Key words: *invariants, deserialization, transformation, mapping, fact.*

Вступ. У сучасних інформаційних системах дані дедалі частіше розподіляються між різними платформами, що відрізняються моделями зберігання, форматами та логікою доступу. Зростання обсягів інформації, динамічність навантажень і поява гібридних архітектур створюють ситуацію, коли класичні підходи до обробки та перенесення даних втрачають ефективність. У таких умовах стає актуальною проблема забезпечення узгодженості та цілісності даних при зміні структури або місця їх розташування.

Завдання ускладнюється тим, що сучасні сховища відрізняються не лише технологічно, а й семантично – те, що в одній системі описується таблицею, в іншій може бути деревом або графом зв'язків. Це унеможливлює пряме копіювання без ризику втрати змісту, типів або зв'язків між елементами даних. Водночас автоматизація таких процесів стає критично необхідною: обсяг даних перевищує межі ручного управління, а вимоги до швидкості, відмовостійкості та відповідності стандартам дедалі зростають.

Проблема полягає у створенні методів, здатних забезпечити коректний перехід між різними структурами зберігання без втрати логічної еквівалентності. Це передбачає не лише технічну трансформацію, а й формальне збереження змісту, обмежень і властивостей даних, що є ключовим для побудови надійних систем у середовищах із високою змінністю та складною інфраструктурою.

Аналіз останніх досліджень і публікацій. У науковому просторі сьогодення представлено значну кількість досліджень, спрямованих на вдосконалення процесів міграції, інтеграції та узгодження даних у багатомодельних і розподілених системах зберігання. Так, у дослідженні Й. Казанавічюса, Д. Мажейки, Д. Калібатени [5] представлено підхід до переходу від монолітної архітектури до мікросервісної з використанням поліглотної персистентності як основи для багатомодельного зберігання даних. Автори розглядають проблему розподілу даних між незалежними сервісами, де кожен має власне сховище, і пропонують методику міграції, що враховує різноманітність моделей даних та особливості взаємодії між компонентами. Практичну реалізацію проведено у вигляді перевірки концепції (proof-of-concept) для бази даних головного фреймворку, яку перетворено на багатомодельну структуру. Оцінювання ефективності здійснено за якісними характеристиками ISO / IEC 25012:2008, зокрема цілісністю, зрозумілістю, доступністю та переносимістю даних. Результати засвідчили, що перехід до поліглотної персистентності підвищує узгодженість і надійність системи, водночас створюючи основу для подальшої автоматизації процесів міграції в умовах складних розподілених середовищ.

У свою чергу М. Перетятко, М. Широкопетлевою, Н. Лісною [12] досліджено проблему підтримки міграції даних між реляційними та документно-орієнтованими моделями зберігання, що є актуальною у контексті переходу до гібридних систем оброблення даних. Відзначені науковці здійснили аналітичний огляд наявних стратегій та методів перенесення інформації, виокремили обмеження традиційних підходів і розробили власний алгоритм гетерогенної модельно-неоднорідної міграції. Особливу увагу приділено застосуванню реляційної алгебри та теорії множин для формалізації процесу перетворення структур і запитів баз даних. Запропонована модель дозволяє узгоджувати логічні зв'язки між сутностями під час переходу з реляційної у документну схему, що знижує ризики втрати цілісності та

підвищує відповідність даних бізнес-запитам. Представлений алгоритм супроводжується математичним описом і UML-діаграмою, що демонструє послідовність кроків формування колекцій у цільовому сховищі. Результати експериментальної перевірки підтвердили працездатність підходу, його здатність забезпечувати повне відтворення інформаційних структур і придатність для практичного використання у реальних сценаріях міграції.

Проблему формалізації процесів інтеграції та міграції даних у багатомодельних і полісторових системах розглянуто В. Уотілою та Дж. Лу [14]. Автори підкреслюють, що сучасні підходи до трансформації даних і схем є фрагментованими та залежать від конкретних доменів, через що відсутня єдина теоретична база. У роботі запропоновано категорно-теоретичну основу для реляційних, графових та ієрархічних моделей, де кожен екземпляр даних подається у вигляді функтора, а перетворення – на основі підняття Кана, що описує універсальну властивість відповідності між схемою та даними. Такий підхід дозволяє формально відобразити процеси трансформації на рівні як структур, так і значень, забезпечуючи узгодженість і логічну цілісність даних у межах різних моделей. Результатом дослідження стало створення єдиного теоретичного каркасу, здатного об'єднати підходи до опису міграцій у гетерогенних середовищах і слугувати основою для подальшої автоматизації процесів узгодження схем і даних.

П. Кукарас [7] здійснив системний огляд сучасних методів інтеграції та зберігання даних у гетерогенних аналітичних середовищах. Розглянуто архітектури, що поєднують традиційні, документоорієнтовані та розподілені системи зберігання, орієнтовані на узгоджене управління великими обсягами інформації. Запропоновано класифікацію, яка поєднує механізми узгодження схем, семантичного збагачення та керування метаданими з рівнями фізичного зберігання і виконання запитів. Особливу увагу приділено забезпеченню масштабованості, узгодженості й відтворюваності даних у складних аналітичних системах. Показано роль онтологічних описів і контролю походження даних для підтримання прозорості, керованості та відтворюваності обчислень. Узагальнені результати демонструють переваги інтегрованих підходів для корпоративних, наукових і державних інформаційних систем, окреслюючи напрями розвитку багаторівневих платформ із підвищеною стійкістю, узгодженістю та довірою до даних.

Наявність значної кількості досліджень у сфері інтеграції, перетворення та узгодження даних у гетерогенних середовищах не виключає необхідності подальшого наукового вивчення питання створення цілісного підходу, що об'єднує формальні моделі представлення, перевірку інваріантів і автоматизоване керування міграцією між різнорідними сховищами.

Постановка завдання. Мета дослідження полягає у розробці концепції та архітектури системи автоматизованої міграції даних, здатної забезпечувати узгодженість, цілісність і еквівалентність інформації під час переходу між різнорідними моделями зберігання в умовах багатоваріантної персистентності.

Виклад основного матеріалу дослідження. Запропонований метод ґрунтується на багатоваріантній моделі зберігання, у якій факт – одна й та сама одиниця інформації – може існувати в кількох еквівалентних представленнях у різних сховищах [15]. Автоматизація міграції забезпечується формалізованими перетвореннями між цими представленнями, що зберігають інваріанти даних [11].

Метод базується на формальній моделі факту, інваріантів і відображень, які визначають логіку процесу міграції [13]. Факт розглядається як абстрактна одиниця даних, інваріантна до способу фізичного зберігання. Формально:

$$\mathrm{Fact} := \mathrm{id}, \mathcal{P}, \mathcal{V}, C, \Pi, \tag{1}$$

де id – стабільний ідентифікатор факту у глобальному просторі ідентифікації; $\mathcal{P}$ – корисне навантаження (payload) у канонічній логічній моделі, задане як елемент алгебраїчного типу або як структура $(\mathcal{A}, \mathcal{R})$ із атрибутами та відношеннями; $\mathcal{V}$ – версійний контекст: множина міток часу або редакцій (t, r), що відстежують еволюцію факту; C – множина доменних обмежень (Constraints) та інваріантів валідності (типи, ключі, кратність зв'язків між компонентами структури, логічні предикати); Π – метадані походження (provenance): джерело, авторство, довірчий рівень, підписи й хеші.

На рівні фізичного зберігання факт існує у вигляді свого варіанта представлення. Припускається, що для сховища S (табличне, документне, ключ-значення, графове тощо) та його моделі даних M_S діє логіка представлення фактів:

$$\mathrm{Rep}_S(\mathrm{Fact}) := S, \mathrm{schema}_S, \mathrm{enc}_S, \Theta_S, \tag{2}$$

де $F \in Fact$ – конкретний факт, schema_S – конкретна схема в межах моделі M_S (наприклад, набір таблиць і ключів, JSON-схема або тип вузлів чи ребер); enc_S – спосіб серіалізації $\mathcal{P}$ у структури M_S (включно з фрагментацією на кілька об'єктів, денормалізацією, індексами); Θ_S – експлуатаційні

параметри та політики: індексація, TTL, класи збереженості, шифрування, механізми узгодженості, транзакційність.

Наявність варіанту представлення задається парою тотальних функцій:

$$E_S : \text{Fact} \to \text{Rep}_S,\ D_S : \text{Rep}_S \to \text{Fact}, \tag{3}$$

де E_S – серіалізація у S, D_S – десеріалізація у канонічну логічну модель. Для коректних представлень має виконуватись умова семантичної ідемпотентності:

$$D_S\left(E_S(F)\right) \equiv_C F, \tag{4}$$

що гарантує логічну еквівалентність відновленого факту до вихідного відносно його інваріантів C. Допускаються незначні синтаксичні відхилення, які не впливають на зміст (наприклад, порядок полів чи формат дат), за умови збереження всіх семантичних властивостей, визначених C.

Багатоваріантність означає, що для фіксованого F може існувати індексована множина його представлень $\left\{R_{S_i}\right\}_{i\in I}$, де кожне R_{S_i} відповідає певному сховищу S_i з різними експлуатаційними профілями Θ_S (латентність, вартість, довірчі гарантії), які залишаються еквівалентними за змістом [3]. Для будь-яких двох представлень з цього сімейства визначається відношення семантичної еквівалентності:

$$R_{S_1} \sim R_{S_2} \Leftrightarrow D_{S_1}\left(R_{S_1}\right) \equiv_C D_{S_2}\left(R_{S_2}\right) \equiv_C F. \tag{5}$$

Ідентичність факту зберігається через його унікальний ідентифікатор id і передається у варіант представлення через ключі або мітки в схемі schema_S. Можливі два випадки: фрагментація та композиція.

Під час фрагментації операція $E_S(F)$ породжує множину фізичних об'єктів $\{ok\}$ із посиланнями між ними. При композиції матеріалізація складного факту агрегує підфакти з власними ідентифікаторами. В обох випадках enc_S має містити механізм відстеження відповідності між id та $\{ok\}$ для коректної десеріалізації D_S [2]. Поза фізичним рівнем представлення, факт описується у контексті $\mathcal{V}$ і П. Представлення повинно зберігати мінімально необхідні мітки версій та часу, щоб відтворювати стан на момент t, та атрибути довіри, потрібні для перевірки автентичності та цілісності [8]. Додатково, типізація фактів визначається канонічними доменами значень і предикатами з множини C, які задають базову логічну структуру даних. Будь-яке представлення повинно мати тотальне відображення доменів у підтримувані типи моделі M_S із визначеною політикою деградації, наприклад із підвищенням розрядності типу int до decimal або з нормалізацією часових поясів [6]. Також необхідно фіксувати стратегії для значень, що не вкладаються в тип системи S: квантизацію, кодування або використання додаткових індексів довідників. Це гарантує збереження відновлюваності при десеріалізації D_S.

Наведені визначення формують основу подальшої формалізації відповідності між моделями даних, інваріантів та коректних перетворень у багатосховищному середовищі [10]. Формалізація відповідності між моделями даних передбачає_структурування для будь-яких двох різнорідних моделей M_i як багатосортних сигнатур із визначеним набором типів, операцій і предикатів. Нехай $\mathcal{T}_i$ є множиною типів у моделі M_i, а K_i є множиною схематичних структур. Формально відповідність задається через пару відображень:

$$\Phi_{\mathcal{T}} : \mathcal{T}_1 \to \mathcal{T}_2,\ \Phi_K : K_1 \to \wp(K_2), \tag{6}$$

де $\Phi_{\mathcal{T}}$ – задає узгодження типів, Φ_K – задає узгодження структур, $\wp(K_2)$ – позначає множину всіх підмножин K_2, що відображає можливість фрагментації, коли елементу K_1 відповідає множина структур K_2 [1].

На рівні семантики відображення Φ визначає схему перетворення фактів, у якій $\Phi_{\mathcal{T}}$ гарантує, що домени значень у M_2 здатні виразити ті ж дані, що і у M_1, а Φ_K задає спосіб матеріалізації цих значень у фізичній структурі.

Для формалізації перетворень на рівні схеми вводиться композиція операцій:

$$\Psi := \psi^{(1)} \circ \psi^{(2)} \circ \cdots \circ \psi^{(k)}, \tag{7}$$

де $\psi^{(j)}$ є атомарними базовими перетвореннями (проєкція, агрегація, розгортання вкладеності, нормалізація, денормалізація, індукція зв'язків через ключі / ідентифікатори). На цьому рівні схема перетворення розглядається як морфізм у категорії схем, який відображає перехід від M_1 до M_2 [9].

Коректність відображення Φ та відповідної схеми трансформацій Ψ визначається трьома взаємопов'язаними вимогами. По-перше, відображення має бути тотальним, тобто всі типові та структурні елементи кожної моделі M_i повинні мати відповідники принаймні в одній іншій моделі. По-друге, воно має забезпечувати часткову оборотність, за якої існує відображення Φ^{-1}, що дозволяє відновити вихідний факт із перетвореної структури. По-третє, трансформація має бути безвтратною, тобто композиція ψ не може призводити до втрати інформації з $\mathcal{P}$ і зобов'язана зберігати ізоморфізм між відповідними семантичними просторами моделей M_i [4]. Уведені визначення факту, його представлень і відображень формують основу формальної моделі, що задає межі допустимих перетворень між

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

різними схемами зберігання. Межі допустимих перетворень задаються набором інваріантів, кожен з яких визначає рівень збереження структури, типів або семантики, необхідний для коректного відновлення змісту з множини представлень.

Структурні інваріанти визначають стабільність топології даних і внутрішню організацію корисного навантаження. Вони фіксують збереження вкладеності, порядку та кількісних співвідношень елементів під час переходу між моделями, підтримуючи незмінність відношень кратності та рівнів ієрархії. Це унеможливлює спрощення або сплющення схеми, що порушує структурну інваріантність. Типові інваріанти забезпечують збереження доменної природи значень та їх типову узгодженість у процесі трансформацій. Для кожного атрибуту канонічного навантаження фіксуються допустимі межі типів і доменів, що унеможливлює втрату семантичних властивостей під час матеріалізації у цільове сховище. Вони гарантують валідність і коректну інтерпретацію значень після десеріалізації. Семантичні інваріанти визначають збереження семантичного змісту_факту незалежно від моделі зберігання. Логічна еквівалентність задається предикатами та обмеженнями C і гарантує, що після будь-якої трансформації факт зберігає ті самі істинні твердження та допустимі інтерпретації. Таким чином, різні форми матеріалізації залишаються еквівалентними за змістом, попри відмінності у вартості, латентності чи доступності.

Для забезпечення дотримання цих інваріантів під час фактичних перетворень даних використовується – mapping DSL (mapping Domain-Specific Language – предметно-орієнтована мова опису трансформації). Формальна семантика цієї мови ґрунтується на операціях над канонічною моделлю факту, де кожен вираз є тотальним відображенням над графом або деревом структур корисного навантаження. Вирази строго типізовані, а операції трактуються як морфізми над структурованими даними, що визначають формальні перетворення. DSL підтримує кілька класів виразів, серед яких атомарні операнди, що відповідають канонічним атрибутам або підструктурам. Композиційні вузли, які описують вкладені набори атрибутів чи підфактів, та операторні конструкції, що формують композиції над атомарними й складеними елементами. Усі вирази мають фіксовану сигнатуру входів і виходів, а їхня семантика визначається відображенням.

Мова включає базовий набір примітивів, які утворюють повний операційний базис і дозволяють будувати довільні тотальні відображення між моделями даних із дотриманням інваріантів:

– project – вибір підмножини атрибутів або полів структури;
– embed – вбудовування підструктури у іншу структуру;
– flatten – перетворення вкладеності у відносну чи табличну форму з виявленням неявної кратності;
– join – поєднання структур за ключами або канонічними ідентифікаторами;
– keyify – індукція ключа або сурогатного ключа (pseudo-key) для фрагментації чи реідентифікації компонентів;
– edgeify – утворення семантичного ребра або відношення між двома сутностями для схем графової моделі.

На етапі валідації виразів DSL застосовуються формальні правила, що забезпечують коректність перетворення перед виконанням. Операції дозволено лише для валідних типів операндів, а тип результату визначається детерміновано за сигнатурою оператора. Композиції допускаються, якщо послідовність не порушує тотальності та збереження доменів. Для складених виразів виконується статична перевірка відсутності немодельованих проєкцій або неопрацьованих атрибутів, що унеможливлює втрату даних, появу неінтерпретованих значень або порушення відновлюваності факту. Мова опису трансформацій є незалежною від формату зберігання: вона визначає уніфікований синтаксис, а семантика реалізується через інтерпретацію примітивів відносно цільової моделі даних. Трансформації DSL не має окремих діалектів – конструкції flatten або embed зберігають однаковий логічний зміст, змінюючи лише спосіб матеріалізації у SQL-документних або ж графових моделях. Базовий механізм реалізації забезпечує інтерпретатор DSL, що інстанціює примітиви у конкретні елементи моделі M_S.

У табличній моделі операції DSL реалізуються як проєкції, введення або нормалізація таблиць за ключами. Ієрархічні вузли подаються окремими таблицями або неатомарними полями з фіксованою кратністю. Оператори join та keyify інтерпретуються відповідно як поєднання за ключами та генерація первинних або сурогатних ідентифікаторів.

У документній моделі (деревоподібній структурі) контейнери DSL відображаються на вкладені об'єкти або масиви. Оператор embed реалізує безпосереднє вкладення підструктур, тоді як flatten – їх розгортання у табличні фрагменти, якщо цього вимагає схема матеріалізації. DSL визначає межі вкладеності та правила винесення фрагментів у зовнішні структури.

У моделі DSL ключ-значення представляє факт як простір ключів, де keyify генерує ключ або їх множину, а корисне навантаження факту зберігається у значенні. Значення може бути вкладеним: у сховищах типу ключ-значення допускається формат JSON або бінарний blob. У цьому випадку DSL визначає не структуру, а стратегію серіалізації та декодування. Оператор flatten може створювати кілька записів ключ-значення для одного факту.

У графовій моделі DSL визначає вузли та ребра. Оператор edgeify встановлює відношення, типи вузлів формуються з атрибутів і підструктур embed, а join задає відповідності між вузлами за ключами. Графова матеріалізація індукується композицією операторів DSL, що визначають, які фрагменти факту стають вузлами, які стають ребрами, і які семантичні мітки отримують відношення.

Гарантії коректності в трансформації DSL визначають кожну трансформацію як тотальну та однозначно визначену функцію. Перший рівень – для будь-якого валідного вхідного факту результат має бути єдиним, а недетермінізм виключено. Така детермінованість забезпечує передбачуваність і стабільність транслятора в усіх цільових моделях – табличних, документних чи типу ключ-значення. Другий рівень гарантій забезпечує статичний аналіз, що виконується до матеріалізації. Під час розбору виразів DSL перевіряється збереження структурних, типових і семантичних інваріантів; якщо хоча б один із них не може бути гарантований, трансформація блокується. Аналіз охоплює не тільки перевірку типів, але й виявлення потенційних проєкцій, які вилучають атрибути корисного навантаження, або композицій, що спричиняють втрату кратності чи неідентифіковану фрагментацію. Третій рівень – політика відхилення при відсутності гарантій збереження інваріантів. У разі неоднозначності або ризику семантичної втрати трансформація відхиляється ще до матеріалізації. Такий механізм фільтрує небезпечні перетворення, запобігаючи порушенню логічної цілісності факту навіть за формально можливої серіалізації.

На основі визначеної мови трансформацій та гарантій збереження інваріантів система отримує можливість не лише виконувати коректні перетворення, а й автоматично приймати рішення про міграцію фактів між сховищами залежно від поточного операційного контексту. Рішення про міграцію реалізуються через систему тригерів, які визначають перехід системи від пасивного багатоваріантного стану до активного переміщення фактів між сховищами. Один тип тригерів пов'язаний із навантаженням: моніторингом CPU, IO, IOPS, черг і рівня конкуренції за ресурси (contention). Якщо модель зберігання демонструє ознаки деградації продуктивності, система ініціює міграцію до моделі, здатної ефективніше обробляти поточний профіль навантаження. Другий тип тригерів ґрунтується на показниках затримки. Якщо поточне сховище перевищує межі умов SLA (Service Level Agreement), що передбачає граничні значення для операцій читання чи запису, тоді система ініціює міграцію до моделі з нижчими накладними витратами. Це забезпечує адаптивну зміну профілів зберігання відповідно до домінуючих типів запитів у певний період. Третій тип тригерів враховує вартісні характеристики. Оцінювання вартості (cost model) оцінює реальну ціну операцій – зберігання, мережевого трафіку та доступу до даних. Якщо прогнозована вартість обробки або зберігання для певного набору фактів перевищує оптимальний рівень, система ініціює міграцію для мінімізації сукупних витрат. Окремий клас тригерів формується AI-агентом, який приймає рішення про міграцію відповідно до політики, побудованої на основі RL або EML. Така політика враховує продуктивність, вартість, патерни доступу, ризики та інші контекстні фактори, що дає змогу системі переходити між різними сховищами не лише реактивно, а й проактивно – за прогнозованими сценаріями використання.

Для побудови політики прийняття рішень використовується формальна модель середовища у вигляді трійки $\mathcal{S}t, \mathcal{A}c, \mathcal{R}ew$, де $\mathcal{S}t$ – простір станів, що відображає поточний профіль навантаження, вартості та стабільності сховищ; $\mathcal{A}c$ – множина можливих дій (переміщення фактів, зміна схеми матеріалізації, призупинення міграції); $\mathcal{R}ew$ – функція винагороди,, що визначає якість переходу між станами. Окремий стан середовища позначається $st \in St$, дія агента – $ac \in Ac$, а результатом виконання дії є новий стан $st' \in St$ і відповідна винагорода $rew = Rew(st, ac)$. Таким чином, кожна взаємодія агента із середовищем описується четвіркою $\langle st, ac, rew, st' \rangle$.

Функція винагороди має вигляд:

$$\mathcal{R}ew = w_1 \text{"}\,\text{Perf} - w_2 \text{"}\,\text{Cost} - w_3 \text{"}\,\text{Lat}, \tag{8}$$

де $\text{"}\,\text{Perf}$ – зміна продуктивності, $\text{"}\,\text{Cost}$ – зміна вартості, $\text{"}\,\text{Lat}$ – зміна затримки. Вагові коефіцієнти w_1, w_2, w_3 визначають пріоритети політики.

На основі цієї функції визначається оптимальна політика $\pi^*(\mathcal{S}t)$, що максимізує очікувану винагороду:

$$\pi^*(\mathcal{S}t) = \arg\max_{\pi} \mathcal{E}[\mathcal{R}ew^t \mid \pi], \tag{9}$$

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

Оптимальна політика може набуватись еволюційним чи підкріплювальним навчанням (RL, EML), що дозволяє системі динамічно пристосовуватись до змінних умов. AI-агенти навчаються в процесі експлуатації системи, формуючи політику міграції на основі накопичених метрик продуктивності, вартості та стабільності. Їхнє навчання здійснюється за принципами підкріплювального або еволюційного машинного навчання, що дозволяє адаптувати поведінку до змін операційного середовища.

Для оцінки дій агент використовує модель цінності стану $U(st)$ або модель цінності дії $Q(st,ac)$, яка апроксимується функцією з параметрами θ (нейронна або регресійна модель). Під час експлуатації агент накопичує досвід $\langle\ st,ac,rew,st'\ \rangle$ і коригує θ за правилом:

$$\theta_{t+1} = \theta_t + \alpha\left(\rho + \gamma \max_{ac'} Q(st',ac') - Q(st,ac)\right)\nabla_\theta Q(st,ac), \quad (10)$$

де α – швидкість навчання, γ – коефіцієнт дисконтування. Така адаптація дає змогу AI-агенту накопичувати емпіричну політику оптимальної міграції без явного програмування правил.

Сам процес міграції реалізується як формальна трирівнева послідовність, що починається з планування та завершується валідацією коректності нової матеріалізації. На етапі планування система визначає цільову модель зберігання та конкретне сховище на основі активованих тригерів. На цьому етапі метрики та політики (затримка, вартість, пропускна здатність) агрегуються у конкретне рішення щодо формату перенесення даних. Планування також охоплює генерацію набору правил трансформації в термінах DSL, які відтворюють канонічний факт у новій матеріалізації. На етапі виконання відбувається безпосереднє застосування опису трансформацій. Трансформація DSL компілюється у план виконання, що формує послідовність операції для окремих фрагментів представлення. Під час виконання зміна логіки перетворень не допускається. Виконання може бути потоковим або пакетним залежно від розміру фактів і характеристик цільового стору. Завершальна фаза – валідація, що виконується під час виконання, як перевірка збереження інваріантів на вибірковій підмножині мігрованих фактів. Такий контроль мінімізує накладні витрати й водночас дає змогу виявляти порушення семантичної еквівалентності.

У разі відхилень міграція може бути відкладена, призупинена або повернена до попередньої матеріалізації за компенсаційними стратегіями. Завершення процесу допускається лише після підтвердження семантичної коректності.

Для забезпечення керованого та безпечного виконання автоматизованої міграції використовується механізм відкату (rollback) і компенсацій. Його основою є журнал фіксації операцій (commit log), що фіксує всі кроки, виконані під час трансформації. Журнал містить дані про переміщені факти, згенеровані ключі, створені структури у цільовому сховищі, а також часові й версіонні мітки. Наявність журналу фіксації операцій дає змогу відтворити порядок дій і визначити фрагменти даних, які можуть потребувати відкату.

Якщо перетворення є оборотними, відкат виконується шляхом застосування оберненої трансформації: за визначеною парою E_S і D_S можливо відновити факт у попередній матеріалізації без втрати інформації. Це дає змогу скасувати наслідки міграції, якщо після валідації або під час експлуатації виявлено погіршення характеристик чи порушення роботи. Якщо ж перетворення не є повністю оборотним, (наприклад, при агрегаціях або редукції вкладеності), застосовується компенсаційна трансформація (compensating mapping) – набір правил для відновлення логічної коректності факту за допомогою додаткових операцій. Навіть у разі часткової невідворотності перетворення система здатна привести факт до стану, що задовольняє семантичні інваріанти й готовий до повторної матеріалізації в іншому сховищі.

Описані вище принципи реалізовано у прототипі, архітектура якого містить шість компонентів, які виконують різні ролі у повному життєвому циклі міграції.

Модуль перевірки інваріантів (Invariant engine) виконує формальну перевірку інваріантів, використовуючи визначення типів, структурних залежностей і семантичних обмежень, щоб оцінити застосовність DSL-перетворень у заданому контексті. Він працює як під час компіляції, так і під час валідації вибіркових підмножин даних після міграції, єдиним джерелом достовірності щодо збереження смислової еквівалентності.

Компілятор трансформації (Mapping compiler) транслює конструкції DSL у виконувану програму міграційного рушія. Під час компіляції створюється граф кроків, де вузли відповідають окремим перетворенням, а ребра задають їх залежності. Він інтерпретує примітиви DSL відносно цільової моделі зберігання, генеруючи інструкції, адаптовані до SQL, документних, ключ-значення та графових систем.

Рис. 1. Архітектура автоматизованої багатоваріантної міграції даних

Джерело: власна розробка автора.

Виконавчий рушій міграції (Migration executor) відповідає за виконання трансформацій, сформованих компілятором. Він реалізує конвеєр (pipeline) виконання, відстежує проміжний стан і веде журнал фіксації операцій. Migration executor контролює транзакційність у межах семантики цільового стору та ініціює відкат або компенсаційне відображення, якщо валідація виявляє порушення інваріантів. Він поєднує операційну функцію з механізмом забезпечення безпеки даних під час міграції.

Інтеграція з наявними сховищами здійснюється за допомогою адаптерів, що інкапсулюють протоколи доступу, транзакційну модель і особливості API конкретної СУБД. Вони забезпечують уніфікований інтерфейс для операцій читання, запису, ізоляції транзакцій і виконання запитів, приховуючи відмінності між табличними, документними, ключ-значення та графовими системами. Другий компонент інтеграції – екстракція схеми (schema extraction), процес побудови логічного представлення структур даних у цільовому сховищі. Система отримує інформацію про типи колонок, вкладеність документів, формати ідентифікаторів, схему графових вузлів і ребер. Екстракція схеми дає модулю перевірки інваріантів змогу зіставляти елементи корисного навантаження з конкретними структурами стору та виявляти невідповідності між формальною моделлю факту й реальною матеріалізацією. Останній компонент, а саме виявлення можливостей (capabilities discovery), яке визначає операційні можливості цільового сховища – транзакційність, підтримку індексів, лічильники з атомарними операціями (atomic counters), з'єднання на боці сервера (server-side join), діапазонні запити (range queries) та підтримку складних фільтрів. На основі цих даних компілятор трансформації обирає стратегію реалізації операторів DSL; інтерпретація трансформацій адаптується до можливостей сховища при збереженні семантичних інваріантів.

Після виконання інтеграційних процедур система надає зовнішнім AI-агентам формалізований інтерфейс взаємодії – API (Application Programming Interface – інтерфейс прикладного програмування). Він забезпечує доступ до сервісів міграції, оцінювання вартості та перевірки інваріантів без прямого втручання у внутрішню семантику системи, а також забезпечує зворотний канал передачі метрик (зміни продуктивності, вартості, латентності) та збереження епізодів $\langle\ st, ac, rew, st'\ \rangle$ у базі досвіду. Це робить можливим безперервне навчання політики без зупинки системи та дозволяє формувати узагальнену модель $\pi^*(st)$ у розподіленому середовищі кількох агентів.

Сервіс міграції обробляє запити зовнішніх агентів, приймаючи параметри набору фактів, цільову модель і бажані операційні властивості. У результаті система повертає декларативну відповідь із підтвердженням або поясненням причини відмови, наприклад, неможливістю гарантувати певний інваріант.

Сервіс оцінювання вартості виконує симуляцію витрат перенесення фактів у різні цільові моделі. Він обчислює прогнозовані показники часу, ресурсів і вартості, що дає змогу оптимізаторам порівнювати сценарії та формувати політики мінімізації сукупного TCO (Total Cost of Ownership – Сукупна вартість володіння) або підвищення QoS (Quality of Service – якість обслуговування).

Сервіс перевірки інваріантів дозволяє агентам передавати опис трансформації разом із вибраними фактами й отримувати формальний висновок про відповідність інваріантам. Такий механізм створює замкнений цикл, у якому AI-система може моделювати, валідувати та лише після цього виконувати операції, не порушуючи формальних гарантій системи.

Для оцінювання було визначено два класи сценаріїв міграції, що відрізнялись складністю та моделями даних. Перший сценарій – простий перехід з моделі SQL до документної. У ньому факт, матеріалізований у табличну структуру з фіксованими ключами та первинними зв'язками, переносився в документну модель із вкладеною ієрархією. Перевірялося, чи коректно виконується операція flatten і чи зберігається кратність після вкладення підструктур. Другий сценарій – складний перехід із моделі

ключ-значення до графової. Тут факт спочатку був представлений у вигляді множини ключів зі значеннями та денормалізованою структурою у форматі blob. У графовій моделі виконувалося edgeify та розділення корисного навантаження на вузли й ребра. Перевірялося, чи не виникає неоднозначності під час індукції ключів і чи забезпечується відтворюваність відношень між компонентами.

Для оцінювання запропонованого методу використано три набори фактів:

1. D1 (synthetic-hier) – 200 тис. фактів із середнім payload 2.4 KB, із вкладеністю до трьох рівнів і комбінацією зв'язків 1: N та N: M через проміжні таблиці.

2. D2 (synthetic-kv) – 150 тис. фактів зі середнім payload 3.1 KB у JSON-blob; ключі частково колізійні (0.2%), передбачено три еволюції схем для перевірки стійкості до змін структури.

3. D3 (stress mix) – 1 млн фактів: 70% D1 + 30% D2, для перевірки масштабованості.

Експерименти проводилися на кластері з трьох вузлів (2×CPU 12C / 24T, 128 GB RAM, NVMe 3.2 GB / s, 25 GbE).

Як цільові сховища використовувалися: реляційне сховище з підтримкою транзакцій та індексів, документне сховище з вкладеними JSON-деревами та індексацією масивів, сховище типу ключ-значення з підтримкою TTL та атомарних лічильників, а також графова база даних із вузлами, ребрами та індексацією за мітками. Для виконання міграції застосовувався розроблений прототип, що включав модуль перевірки інваріантів, компілятор трансформацій і виконавчий рушій міграції з моніторингом метрик та журналом фіксації операцій. Процедура вимірювань складалася з кількох етапів: спочатку фіксувався стан вихідного сховища шляхом створення знімка стану, далі компілювався опис трансформацій DSL у план матеріалізації, після чого виконувалися перетворення з пакетним розміром 5,000 фактів і паралелізмом до 8 потоків. Для 5% вибірки кожного пакета здійснювалася перевірка під час роботи. Кожен сценарій повторювали 5 разів із перемішуванням порядку пакетів, а отримані результати усереднювали.

Оцінювання здійснювалось за набором формальних метрик. Першою метрикою був T_{mig} час виконання міграції, що фіксував загальні витрати на матеріалізацію у цільовому сховищі. Другою метрикою був Viol – число порушених інваріантів – вимога передбачала, що це значення має залишатись нульовим для розробленої моделі. Додатковими метриками були накладні витрати на інфраструктурному рівні: пікові CPU%, NET out (GB) – мережевий трафік, та затримка узгодження між представленнями. Ці показники дозволяли оцінити операційні витрати при збільшенні масштабу міграції.

Для порівняння використовувалися два базові підходи. Перший – ручні трансформації без формального контролю інваріантів. Другий – простий процес ETL (Extract, Transform, Load – Витяг, Перетворення та Завантаження) без перевірки інваріантів, у якому дані переносилися механічно. Під час експериментів AI-агенти працювали у режимі спостереження: політика $\pi^*(st)$ була фіксованою, а система лише накопичувала епізоди $\langle st, ac, rew, st' \rangle$ для подальшого навчання. Результати табл. 1–3 відображають початковий цикл роботи без адаптації політики.

Таблиця 1

Порівняння результатів міграції з моделі SQL до документної моделі

Підхід	T_{mig}	Viol	пікові CPU%	NET out (GB)	Затримка узгодження
Зап. метод	812	0	46	9.8	42
Руч. Т.	905	7	44	10.1	67
ETL	668	124	38	9.5	59

Джерело: власна розробка автора.

Таблиця 2

Порівняння результатів міграції зі сховища типу ключ-значення до графової моделі

Підхід	T_{mig}	Viol	пікові CPU%	NET out (GB)	Затримка узгодження
Зап. метод	1196	0	52	12.7	71
Руч. Т.	1318	23	49	12.5	104
ETL	1034	213	41	12.1	97

Джерело: власна розробка автора.

Таблиця 3

Порівняння результатів міграції на стрес-тестовому наборі даних

Сценарій	Підхід	T_{mig}	Viol	пікові CPU%	NET out (GB)	Затримка узгодження
Перший	Зап. метод	3920	0	61	48.2	128
Перший	ETL	3215	627	55	46.9	141
Другий	Зап. метод	5480	0	66	63.7	171
Другий	ETL	4770	981	57	61.8	208

Джерело: власна розробка автора.

Отримані результати (табл. 1–3) вказують на те, що класичний ETL забезпечує найменший час міграції, однак супроводжується найбільшою кількістю порушень інваріантів, особливо у складних сценаріях. Ручні трансформації зменшують кількість помилок, проте поступаються швидкодією. Запропонований метод гарантує відсутність порушень інваріантів (Viol = 0) за незначного збільшення часу міграції, а саме у межах 15–25% відносно ETL. Навчальні агенти на цьому етапі виконують роль моніторингового модуля, що накопичує досвід і формує статистику для подальшого підвищення ефективності рішень у наступних циклах.

Таким чином, запропонований метод забезпечує коректну автоматизовану міграцію даних між різнорідними сховищами, зберігаючи повну семантичну еквівалентність і цілісність фактів. Завдяки формалізованим перетворенням, перевірці інваріантів та механізмам відкату і компенсацій система гарантує надійність і відсутність втрат даних при зміні моделі зберігання. Експерименти підтвердили, що метод підтримує нульовий рівень порушень інваріантів за мінімального приросту часу міграції, демонструючи стабільність і передбачуваність результатів.

Висновки. У ході виконання дослідження послідовно сформовано та реалізовано формальний метод автоматизованої міграції даних між різнорідними сховищами у системах з багатоваріантною персистентністю. Побудовано математичну модель факту, інваріантів і відображень, що визначають межі допустимих перетворень та гарантують семантичну еквівалентність між різними матеріалізаціями даних. Використано предметно орієнтовану мову опису трансформацій, яка забезпечує формальну верифікацію тотальності, типізації та збереження інваріантів під час виконання перетворень. Реалізовано архітектуру системи, що поєднує модулі компіляції, виконання, перевірки інваріантів і механізми відкату та компенсації, а також інтеграцію з різними моделями сховищ через уніфіковані адаптери. Розроблений прототип засвідчив, що система спроможна здійснювати коректну та відмовостійку міграцію даних без втрати цілісності, забезпечуючи нульовий рівень порушень інваріантів при мінімальному зростанні часу виконання. Проведені експерименти вказують стабільність результатів і придатність методу до масштабування. Отримані результати підтверджують ефективність розробленої моделі та відкривають перспективи подальшої адаптації системи через використання навчальних агентів і динамічних політик керування міграцією.

Подальші дослідження можуть бути спрямовані на розвиток адаптивних механізмів керування міграцією, зокрема з використанням навчальних агентів у контурі прийняття рішень. Передбачається дослідити вплив активного навчання політики на скорочення часу, затримок і витрат, а також розробити багатокритеріальні методи оцінки ефективності для динамічного розподілу даних у багатоваріантних середовищах.

Список використаних джерел:

1. Fernández Candel C. J., Sevilla Ruiz D., García-Molina J. J. A unified metamodel for NoSQL and relational databases. *Information Systems*. 2022. Vol. 104. 101898. DOI: https://doi.org/10.1016/j.is.2021.101898

2. Glake D., Kiehn F., Schmidt M., Panse F., Ritter N. Towards polyglot data stores – overview and open research questions. *arXiv preprint*: website. 2022. DOI: https://doi.org/10.48550/arXiv.2204.05779

3. Goch L., Chen S. Data migration in large scale storage systems with varying file sizes. *Proceedings of the 2024 8th International Conference on Algorithms, Computing and Systems (ICACS '24)*. 2025. P. 77–82. DOI: https://doi.org/10.1145/3708597.3708609

4. Hussein A. A. Data migration need, strategy, challenges, methodology, categories, risks, uses with cloud computing, and improvements in its using with cloud using suggested proposed model (DMig 1). *Journal of Information Security*. 2021. Vol. 12. No. 1. P. 49–64. DOI: https://doi.org/10.4236/jis.2021.121004

5. Kazanavičius J., Mažeika D., Kalibatienė D. An Approach to Migrate a Monolith Database into Multi-Model Polyglot Persistence Based on Microservice Architecture: A Case Study for Mainframe Database. *Applied Sciences*. 2022. Vol. 12. No. 12. 6189. DOI: https://doi.org/10.3390/app12126189

6. Kiehn F., Schmidt M., Glake D., Panse F., Wingerath W., Wollmer B., Poppinga M., Ritter N. Polyglot data management: state of the art & open challenges. *Proceedings of the VLDB Endowment*. 2022. Vol. 15. No. 12. P. 3750–3753. DOI: https://doi.org/10.14778/3554821.3554891

7. Koukaras P. Data integration and storage strategies in heterogeneous analytical systems: architectures, methods, and interoperability challenges. *Information*. 2025. Vol. 16. No. 11. 932. DOI: https://doi.org/10.3390/info16110932

8. Koupil P., Holubová I. A unified representation and transformation of multi-model data using category theory. *Journal of Big Data*. 2022. Vol. 9. Article No. 61. DOI: https://doi.org/10.1186/s40537-022-00613-3

9. Lu J., Liu Z. H., Xu P., Zhang C. UDBMS: road to unification for multi-model data management. *arXiv preprint*: website. 2016. DOI: https://doi.org/10.48550/arXiv.1612.08050.

10. Nadal S., Romero O., Abelló A., Vassiliadis P., Vansummeren S. An integration-oriented ontology to govern evolution in big data ecosystems. *Information Systems*. 2018. Vol. 76. P. 68–88. DOI: https://doi.org/10.1016/j.is.2018.01.006

11. Nadig R., Arulchelvan V., Bera R., Shahroodi T., Singh G., Kakolyris A., Sadrosadati M., Park J., Mutlu O. Harmonia: a multi-agent reinforcement learning approach to data placement and migration in hybrid storage systems. *arXiv preprint*: website. 2025. DOI: https://doi.org/10.48550/arXiv.2503.20507

12. Peretiatko M., Shirokopetleva M., Lesna N. Research of methods to support data migration between relational and document data storage models. *Innovative Technologies and Scientific Solutions for Industries*. 2022. No. 2 (20). P. 64–74. DOI: https://doi.org/10.30837/ITSSI.2022.20.064

13. Ramos-Vidal D., Cortiñas A., Luaces M. R., Pedreira O., Saavedra Places Á., Assunção W. K. G. Seamless data migration between database schemas with DAMI-framework: an empirical study on developer experience. *arXiv preprint*: website. 2025. DOI: https://doi.org/10.48550/arXiv.2504.17662

14. Uotila V., Lu J. A formal category theoretical framework for multi-model data transformations. *Lecture Notes in Computer Science*. 2021. Vol. 12921. P. 14–28. DOI: https://doi.org/10.1007/978-3-030-93663-1_2

15. Ye F., Sheng X., Nedjah N., Sun J., Zhang P. A benchmark for performance evaluation of a multi-model database vs. polyglot persistence. *Journal of Database Management*. 2023. Vol. 34. No. 3. P. 20. DOI: https://doi.org/10.4018/JDM.321756

Дата надходження статті: 16.11.2025
Дата прийняття статті: 10.12.2025
Опубліковано: 30.12.2025