 ISSN 2071-1077(print)
ISSN 2663-2411(online

doi: 10.32620/oikit.2026.107.14

УДК 004.65

І. В. Шевченко, О. В. Лучшева, П. О. Лучшев

Дослідження особливостей міграції даних з реляційної в документо-орієнтовану модель з використанням GUI-клієнта Studio 3T

Національний аерокосмічний університет «Харківський авіаційний інститут»

У статті досліджено особливості міграції даних із реляційної бази даних MySQL у документо-орієнтовану базу даних MongoDB. Об’єктом дослідження є процес міграції реляційної моделі даних у документо-орієнтовану з урахуванням структурних відмінностей між SQL- та NoSQL-підходами. Предметом дослідження є методи представлення реляційних зв’язків у MongoDB та інструментальні засоби їх реалізації. Метою статті є аналіз підходів до міграції таблиць і зв’язків типу один-до-одного та один-до-багатьох у структуру документів MongoDB із використанням графічного клієнта Studio 3T. У роботі використано реляційну схему world як приклад джерела даних для міграції у колекції документів MongoDB. Проаналізовано стратегії денормалізації, збереження посилань і вбудовування пов’язаних сутностей у документи, а також використання масивів скалярних значень і масивів вкладених об’єктів. Показано, що вибір моделі зберігання даних у MongoDB залежить від характеру запитів, частоти оновлення інформації та вимог до продуктивності системи. Проведено практичну міграцію даних за допомогою Studio 3T та оцінено можливості цього інструмента для проєктування та тестування структури документів. Отримані результати підтверджують доцільність використання документо-орієнтованих баз даних для систем, що потребують швидкого доступу до пов’язаних даних, і можуть бути використані у навчальних та прикладних задачах проєктування NoSQL-сховищ.

Ключові слова: реляційні БД; документо-орієнтовані БД; MongoDB; GUI-клієнт; Studio 3T; міграція даних.

Вступ

Сучасний цифровий простір характеризується стрімким зростанням обсягів напівструктурованих даних, які генеруються веб-застосунками, мобільними застосунками, IoT-пристроями та системами аналітики в реальному часі. За оцінками аналітиків, обсяг неструктурованих та напівструктурованих даних зростає швидше за структуровані дані приблизно у п’ять разів [1], що створює нові виклики для традиційних реляційних систем управління базами даних. У відповідь на ці виклики NoSQL-рішення, зокрема документо-орієнтовані бази даних, набули широкого поширення у веб-системах завдяки своїй здатності ефективно обробляти динамічні та складні структури даних [2].

Міграція з реляційних баз даних до NoSQL стала типовою практикою для компаній різного масштабу. Відомі кейси включають перехід eBay до MongoDB для управління каталогом товарів [3], міграцію систем аналітики Cisco до документо-орієнтованих сховищ, а також численні випадки модернізації legacy-систем у фінтех-секторі та e-commerce. Ці приклади демонструють не лише технічну можливість такої міграції [4], але й відчутні бізнес-результати: покращення швидкості розроблення, зниження операційних витрат та підвищення гнучкості у відповідь на зміни бізнес-вимог.

Важливим аспектом успішної міграції є вибір відповідних інструментів, які дозволяють не лише автоматизувати процес перенесення даних, але й глибоко дослідити структурні трансформації на етапі проєктування. У цьому контексті

Studio 3T виступає не просто як допоміжний інструмент для виконання технічних операцій, а як повноцінне середовище для аналізу, моделювання та верифікації стратегій міграції, що робить його центральним елементом дослідження процесів трансформації даних між різними парадигмами зберігання.

1. Аналіз публікацій

Рішення про міграцію з реляційної бази до документо-орієнтованої зумовлюється комплексом технічних та бізнесових факторів. Першочерговою причиною часто виступає проблема горизонтальної масштабованості: реляційні бази традиційно масштабуються вертикально, що створює природні обмеження та призводить до експоненційного зростання вартості обладнання. Документо-орієнтовані системи, навпаки, проєктуються з урахуванням розподіленої архітектури та шардингу [5], що дозволяє ефективно масштабуватися горизонтально на commodity-серверах.

Значний вплив має також продуктивність операцій читання у системах із складними зв’язками. У нормалізованих реляційних схемах отримання повної інформації про сутність часто вимагає виконання множинних JOIN-операцій, що за великих обсягів даних призводить до зниження продуктивності. Денормалізована структура документів забезпечує атомарне читання всієї необхідної інформації одним запитом [6], що критично важливо для високонавантажених веб-систем.

Гнучкість схеми даних стає особливо цінною у контексті agile-розроблення програмного продукту з інкрементною та ітеративною моделлю розвитку. У реляційних системах зміна структури даних вимагає виконання зміни схеми, що може призводити до простоїв та ускладнює процес continuous deployment. Документо-орієнтовані бази дозволяють еволюціонувати структуру даних органічно, зберігаючи зворотну сумісність та дозволяючи різним версіям застосунка працювати з одними й тими ж даними.

Додатковим фактором для міграції є природна відповідність між документо-орієнтованою моделлю та об’єктно-орієнтованим кодом сучасних застосунків. Процес встановлення відповідності (mapping) між таблицями та об’єктами (ORM) часто вимагає складних конфігурацій та створює додатковий рівень абстракції, тоді як документи можуть безпосередньо серіалізуватися з об’єктів, спрощуючи архітектуру застосунка.

Процес міграції з реляційної в документо-орієнтовану модель супроводжується низкою концептуальних та технічних викликів. Центральною проблемою є відповідність між нормалізованими таблицями та денормалізованими документами: необхідно прийняти стратегічні рішення щодо того, які пов’язані дані слід вбудувати у документ (embedding), а які зберегти як окремі документи з відповідними посиланнями (referencing). Ці рішення мають критичний вплив на продуктивність, консистентність та складність підтримки системи [7].

Трансформація операції JOIN між реляційними таблицями представляє особливий виклик. У реляційних системах JOIN є фундаментальним механізмом отримання даних з множинних таблиць, тоді як у документо-орієнтованих базах цей підхід є антипатерном. Міграція вимагає ретельного аналізу патернів доступу до даних та реструктуризації схеми таким чином, щоб мінімізувати потребу в операціях lookup або aggregation на рівні бази даних.

Питання транзакцій та цілісності даних також потребує уважного розгляду при міграції. Якщо реляційні бази гарантують ACID-властивості для

багатотабличних транзакцій, то документо-орієнтовані системи традиційно забезпечують атомарність лише на рівні окремого документа. Хоча сучасні версії MongoDB підтримують мультидокументні транзакції, їх використання має обмеження через їх продуктивність, що вимагає переосмислення підходів до забезпечення консистентності даних [2].

Додаткові виклики підчас міграції включають збереження референційної цілісності даних без механізму зовнішніх ключів, міграцію складних constraint'ів та тригерів, а також забезпечення коректності типів даних при переході від жорстко типізованих стовпців до гнучкої JSON-структури. Кожен з цих аспектів вимагає детального планування та верифікації на всіх етапах процесу міграції.

Міграція даних з реляційних баз до MongoDB може здійснюватися різними підходами, кожен з яких має свої переваги та обмеження залежно від масштабу проєкту, складності схеми та бізнес-вимог.

ETL-підхід (Extract, Transform, Load) передбачає використання спеціалізованих платформ, таких як Apache NiFi, Talend або Pentaho Data Integration. Ці інструменти забезпечують візуальне проєктування ланцюгів (pipelines) трансформації даних [8], підтримку складних бізнес-правил та можливість інкрементальної міграції. Проте вони вимагають значних інвестицій у навчання та конфігурацію, що може бути надмірним для проєктів середнього масштабу.

Програмні скрипти на мовах Python, Node.js або Java надають максимальну гнучкість та контроль над процесом міграції [5]. Використовуючи бібліотеки як PyMongo або офіційний MongoDB Node.js driver у поєднанні з SQL-коннекторами, розробники можуть реалізувати складну логіку трансформації, специфічну для конкретного домену. Однак цей підхід вимагає значних зусиль на розроблення, тестування та підтримку коду міграції.

ORM-орієнтовані підходи передбачають використання об'єктно-реляційних перетворювачів (mappers), таких як Mongoose для Node.js або PyMongo з ODM-обгортками. Цей метод дозволяє моделювати цільову структуру документів у термінах об'єктно-орієнтованого коду та використовувати існуючі ORM-інструменти для читання з реляційної бази. Підхід ефективний для застосунків, що вже використовують ORM, але може бути обмеженим для складних трансформацій.

GUI-інструменти, такі як Studio 3T, MongoDB Compass та NoSQLBooster, надають інтуїтивний інтерфейс для міграції даних без необхідності написання коду [4]. Вони особливо ефективні для проєктів середнього розміру, де потрібен баланс між швидкістю впровадження та гнучкістю налаштувань.

2. Мета роботи і постановка завдань

Метою статті є дослідження особливостей реалізації процесу міграції бази даних з реляційної моделі в документо-орієнтовану на прикладі MongoDB з використанням GUI-клієнта Studio 3T, а також аналіз особливостей відображення реляційних зв'язків у документо-орієнтованій моделі даних.

Для досягнення поставленої мети у роботі необхідно розв'язати такі завдання:

- проаналізувати можливості GUI-клієнта Studio 3T щодо міграції даних;
- реалізувати міграцію однієї реляційної таблиці в MongoDB;
- дослідити міграцію реляційних таблиць, пов'язаних зв'язком один-до-одного;

ISSN 2071-1077(print)
ISSN 2663-2411(online

– дослідити міграцію реляційних таблиць, пов’язаних зв’язком один-до-багатьох;

– проаналізувати особливості відображення реляційних зв’язків у документо-орієнтованій моделі даних;

– оцінити переваги та обмеження використання GUI-клієнта Studio 3T для міграції даних.

3. Вибір реляційної схеми даних для реалізації міграцій

Для демонстрації різних видів міграцій виберемо реляційну СКБД MySQL і схему даних world, яка надається розробником MySQL. Схема даних world містить три пов’язані між собою таблиці *city*, *country* і *countylanguage*.

У таблицях 1-3 наведено структури зазначених реляційних таблиць, а на рисунках 1-3 – приклади даних, які зберігаються у відповідних таблицях і які будемо перетворювати для зберігання у MongoDB у вигляді документів.

Таблиця 1

Реляційна таблиця *city*

Стовпець	Детальний опис
ID	INT – первинний ключ (автоінкремент)
Name	CHAR(35) – назва міста
CountryCode	CHAR(3) – код країни (зовнішній ключ до country.Code)
District	CHAR(20) – район / область / штат
Population	INT – чисельність населення міста

Загальна кількість рядків у таблиці *city* складає 4079. Значення NULL не допускається в жодному зі стовпців.

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabol	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilburg	NLD	Noord-Brabant	193238
11	Groningen	NLD	Groningen	172701
12	Breda	NLD	Noord-Brabant	160398
13	Apeldoorn	NLD	Gelderland	153491
14	Nijmegen	NLD	Gelderland	152463
15	Enschede	NLD	Overijssel	149544

Рис. 1. Приклад даних, які зберігаються у таблиці *city*

Таблиця 2

Реляційна таблиця *country*

Стовпець	Детальний опис
Code	CHAR(3) – первинний ключ (ISO-код із трьох літер)
Name	CHAR(52) – назва країни
Continent	ENUM(...) – континент

ISSN 2071-1077(print)
ISSN 2663-2411(online

Продовження таблиці 2

Стовпець	Детальний опис
Region	CHAR(26) – регіон
SurfaceArea	DECIMAL(10,2) – площа території
IndepYear	SMALLINT – рік здобуття незалежності
Population	INT – чисельність населення
LifeExpectancy	DECIMAL(3,1) – середня тривалість життя
GNP	DECIMAL(10,2) – валовий національний продукт
GNPOld	DECIMAL(10,2) – попереднє значення валового національного продукту
LocalName	CHAR(45) – назва країни місцевою мовою
GovernmentForm	CHAR(45) – форма державного правління
HeadOfState	CHAR(60) – глава держави
Capital	INT – ідентифікатор столиці (зовнішній ключ до city.ID)
Code2	CHAR(2) – дволітерний код країни відповідно до стандарту ISO

Загальна кількість рядків у таблиці *country* складає 239. Значення NULL допускається в таких стовпцях: IndepYear, LifeExpectancy, GNP, GNPOld, HeadOfState, Capital.

Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpectancy
ABW	Aruba	North America	Caribbean	193.00	NULL	103000	78.4
AFG	Afghanistan	Asia	Southern and Central Asia	652090.00	1919	22720000	45.9
AGO	Angola	Africa	Central Africa	1246700.00	1975	12878000	38.3
AIA	Anguilla	North America	Caribbean	96.00	NULL	8000	76.1
ALB	Albania	Europe	Southern Europe	28748.00	1912	3401200	71.6
AND	Andorra	Europe	Southern Europe	468.00	1278	78000	83.5
ANT	Netherlands Antilles	North America	Caribbean	800.00	NULL	217000	74.7
ARE	United Arab Emirates	Asia	Middle East	83600.00	1971	2441000	74.1
ARG	Argentina	South America	South America	2780400.00	1816	37032000	75.1
ARM	Armenia	Asia	Middle East	29800.00	1991	3520000	66.4
ASM	American Samoa	Oceania	Polynesia	199.00	NULL	68000	75.1
ATA	Antarctica	Antarctica	Antarctica	13120000.00	NULL	0	NULL
ATF	French Southern ter…	Antarctica	Antarctica	7780.00	NULL	0	NULL
ATG	Antigua and Barbuda	North America	Caribbean	442.00	1981	68000	70.5
AUS	Australia	Oceania	Australia and New Zealand	7741220.00	1901	18886000	79.8

Рис. 2. Приклад даних, які зберігаються у таблиці *country*

Таблиця 3

Реляційна таблиця *countylanguage*

Стовпець	Детальний опис
CountryCode	CHAR(3) – код країни (частина складеного первинного ключа)
Language	CHAR(30) – назва мови (частина складеного первинного ключа)
IsOfficial	ENUM(‘T’,’F’) – ознака офіційного статусу мови
Percentage	DECIMAL(4,1) – відсоток населення, що розмовляє цією мовою

Загальна кількість рядків у таблиці *countylanguage* складає 984. Значення NULL не допускається в жодному зі стовпців.

Детально опишемо зв’язки між таблицями, які в подальшому будемо використовуватися для реалізації міграцій.

Між таблицями *country* і *city* реалізовано зв’язок один-до-одного, який використовується для визначення столиці держави: кожна країна має одну

ISSN 2071-1077(print)
ISSN 2663-2411(online

столицю, одне місто може бути столицею лише однієї країни. Зв’язок реалізовано за допомогою зовнішнього ключа: country.Capital → city.ID.

Між таблицями *country* і *countrylanguage* реалізовано зв’язок один-до-багатьох, який використовується для визначення мов, що використовуються на території відповідних держав: одна країна може мати декілька записів про мови, але кожен такий запис належить лише одній країні. Зв’язок реалізовано за допомогою зовнішнього ключа: countrylanguage.CountryCode → country.Code.

CountryCode	Language	IsOfficial	Percentage
ABW	Dutch	T	5.3
ABW	English	F	9.5
ABW	Papiamento	F	76.7
ABW	Spanish	F	7.4
AFG	Balochi	F	0.9
AFG	Dari	T	32.1
AFG	Pashto	T	52.4
AFG	Turkmenian	F	1.9
AFG	Uzbek	F	8.8
AGO	Ambo	F	2.4
AGO	Chokwe	F	4.2
AGO	Kongo	F	13.2
AGO	Luchazi	F	2.4
AGO	Luimbe-nganguela	F	5.4
AGO	Luvale	F	3.6
AGO	Mbundu	F	21.6
AGO	Nyaneka-nkhumbi	F	5.4
AGO	Ovimbundu	F	37.2

Рис. 3. Приклад даних, які зберігаються у таблиці *countylanguage*

4. Можливості GUI-клієнта Studio 3T для міграції даних

Studio 3T позиціонується як найбільш функціональний GUI-клієнт для MongoDB з розширеними можливостями міграції даних з реляційних систем. На відміну від базових інструментів, таких як MongoDB Compass, Studio 3T надає комплексне рішення для планування, виконання та верифікації процесів міграції даних між різними парадигмами зберігання.

Data Import Wizard є центральним інструментом для міграції даних з реляційних баз до MongoDB. Майстер імпорту підтримує прямий імпорт з найпоширеніших реляційних СУБД: MySQL, PostgreSQL, Microsoft SQL Server та Oracle. З’єднання з реляційними базами здійснюється через JDBC, що забезпечує стандартизований доступ до різних джерел даних без необхідності встановлення додаткових драйверів або бібліотек.

Процес міграції організовано у вигляді послідовних кроків, кожен з яких відповідає за певний аспект такого перетворення:

– встановлення з’єднання з реляційною базою даних (налаштування параметрів підключення, вибір бази даних);

– вибір таблиць для міграції (підтримка як окремих таблиць, так і множинного вибору);

– налаштування відповідності (mapping) між стовпцями реляційних таблиць та полями документів MongoDB;

– визначення цільової колекції MongoDB та параметрів імпорту;

– виконання міграції з можливістю моніторингу прогресу та перегляду логів.

 ISSN 2071-1077(print)
ISSN 2663-2411(online

Schema Mapping Tool надає візуальний інтерфейс для налаштування відповідності між стовпцями реляційних таблиць та полями документів MongoDB. Інструмент дозволяє:

– визначати типи даних для кожного поля документа;

– перейменовувати поля при міграції для відповідності конвенціям іменування MongoDB;

– виключати непотрібні стовпці з процесу міграції;

– налаштовувати трансформацію NULL-значень (пропуск поля, заміна на default-значення, збереження як null);

– встановлювати обробку помилок конвертації типів даних.

Relationship Mapping є ключовою функціональністю, що відрізняє Studio 3T від простих інструментів імпорту. Цей механізм дозволяє трансформувати реляційні зв’язки, реалізовані через зовнішні ключі, у різні стратегії організації даних у MongoDB:

– referencing для збереження посилань між документами (аналог зовнішніх ключів);

– embedding для вбудовування пов’язаних даних безпосередньо у документ;

– array embedding для створення масивів вкладених документів для зв’язків один-до-багатьох.

Для налаштування перетворення зв’язків Relationship Mapping надає візуальний інтерфейс, де користувач може:

– вибрати зовнішній ключ у вихідній таблиці;

– визначити цільову таблицю та відповідний первинний ключ;

– обрати тип перетворення (скалярне значення, об’єкт або масив);

– вказати, які поля з пов’язаної таблиці слід включити у вбудований документ.

Завдяки цим можливостям Studio 3T виступає не лише як інструмент технічного виконання міграції, але й як платформа для дослідження різних стратегій трансформації даних, їх валідації та оптимізації перед фінальним впровадженням у робочому середовищі, в якому програмний продукт фактично буде використовуватися кінцевими користувачами (production-середовищі).

5. Міграція в MongoDB однієї реляційної таблиці

Реалізуємо просту міграцію реляційної таблиці country в локальну MongoDB. У налаштуваннях міграції SQL → MongoDB треба задати Source SQL connection, Target MongoDB connection, а також source table (вибрати таблицю *country*), target database (задати значення *test*), target collection (задати значення *collection_country*) і Insertion mode (залишити режим за замовчуванням).

Поля створеної колекції collection_country повинні мати значення, які будуть повністю відповідати значенням реляційної таблиці.

Загальні налаштування Mappings (рисунок 4), які будемо використовувати у всіх міграціях, будуть полягати у:

– визначенні дій з SQL NULL values (вибрати опцію *Exclude from target document*);

– визначенні дій при виникненні помилки міграції (вибрати опцію *Exclude the field*);

– виключенні з міграції поля *Code* через наявність автоматичного

ISSN 2071-1077(print)
ISSN 2663-2411(online

ідентифікатора документа *_id*, значення якого співпадає зі значенням поля *Code*.

У результаті виконання міграції (Run migration) в MongoDB буде створена колекція *collection_country* з 239 документів, які відповідатимуть 239 рядкам реляційної таблиці country.

Приклад документа створеної колекції *collection_country*:

```
{  "_id" : "ABW",
   "Name" : "Aruba",
   "Continent" : "North America",
   "Region" : "Caribbean",
   "SurfaceArea" : NumberDecimal("193.00"),
   "Population" : NumberInt(103000),
   "LifeExpectancy" : NumberDecimal("78.4"),
   "GNP" : NumberDecimal("828.00"),
   "GNPOld" : NumberDecimal("793.00"),
   "LocalName" : "Aruba",
   "GovernmentForm" : "Nonmetropolitan Territory of The Netherlands",
   "HeadOfState" : "Beatrix",
   "Capital" : NumberInt(129),
   "Code2" : "AW"
}
```

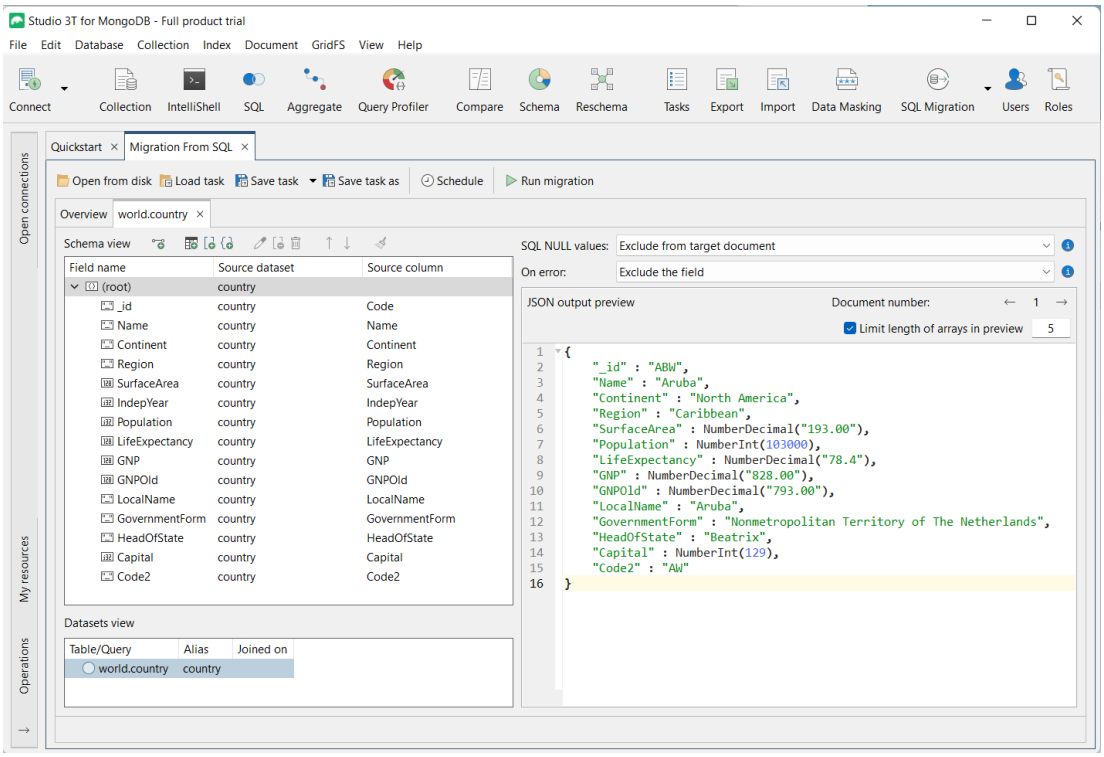

Рис. 4. Попередній перегляд перед міграцією даних

6. Міграція в MongoDB реляційних таблиць, пов'язаних зв'язком один-до-одного

При реалізації простої міграції таблиці *country* в MongoDB у колекцію *collection_country* кожен документ отриманої колекції містить ідентифікатор столиці країни, що відповідає значенню поля *Capital* реляційної таблиці. Для вище наведеного прикладу документу, який відповідає країні Aruba, поле *Capital* має значення NumberInt(129).

Однак Studio 3T дозволяє двома способами налаштувати додавання

ISSN 2071-1077(print)
ISSN 2663-2411(online

замість ідентифікатора столиці інформацію про неї.

Часткове вбудовування (partial embedding)

За допомогою Mappings можна налаштувати додавання назви столиці замість її ідентифікатора.

Така можливість обумовлена наявністю між реляційними таблицями *country* і *city* зв'язку один-до-одного, який реалізується через зовнішній ключ country.Capital, що посилається на первинний ключ city.ID.

У процесі міграції даних цей зв'язок може бути трансформований таким чином: зовнішній ключ country.Capital реляційної таблиці *country* замінюється у документі MongoDB на скалярне поле – назву столиці, яка буде отримана з пов'язаної таблиці *city* за відповідним первинним ключем city.ID.

Додаткові налаштування Mappings (рисунок 5), які треба виконати для поточної міграції, будуть полягати у:

– видаленні запропонованого за замовчуванням поля *Capital*;

– додаванні нового поля *Capital* з реалізацію зв'язка country.Capital → city.ID (тип нового поля String, джерело даних – стовпець city.Name).

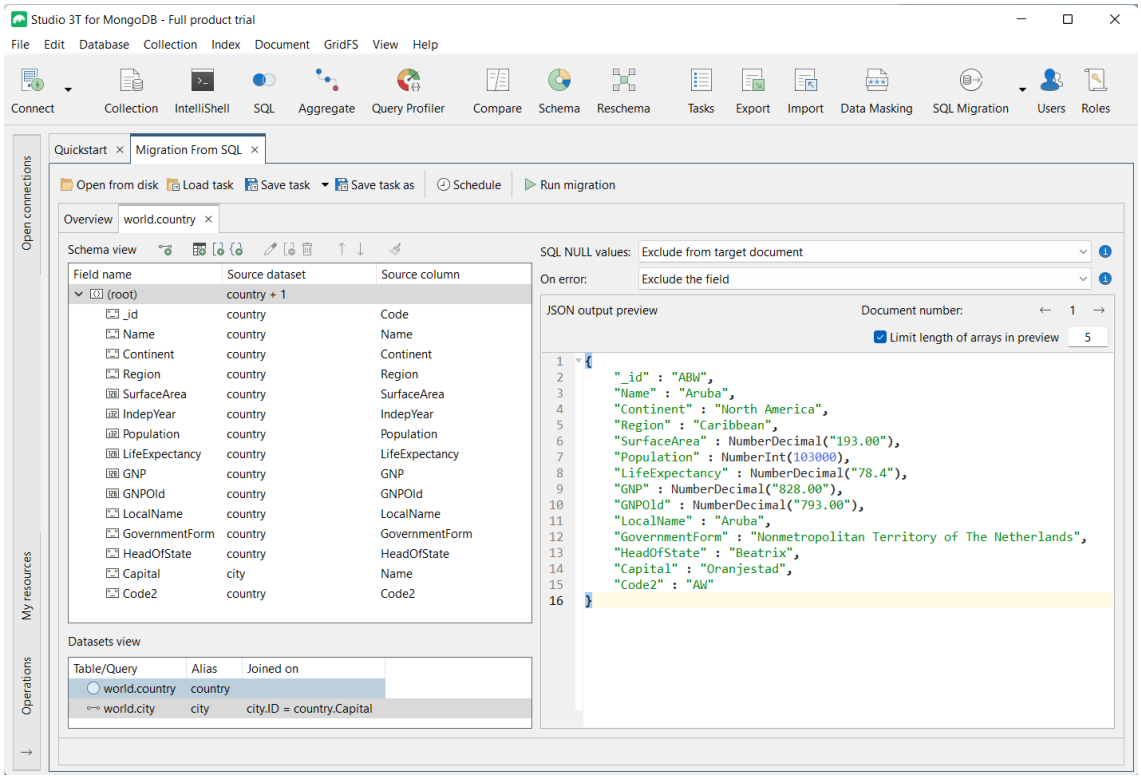

Рис. 5. Попередній перегляд перед міграцією даних

У результаті виконання міграції (Run migration) в MongoDB створена колекція *collection_country* буде містити в одному документі не тільки інформацію про відповідну країну, а також інформацію про назву столиці:

{
"_id" : "ABW",
"Name" : "Aruba",
"Continent" : "North America",
"Region" : "Caribbean",
…
"Capital" : "Oranjestad"

```
}
```

У MySQL для отримання інформації щодо назви столиці для конкретної країни необхідно виконати JOIN-запит до двох реляційних таблиць *country* і *city*.

Повне вбудовування (full embedding)

GUI-клієнт Studio 3T за допомогою інструмента Mappings дозволяє виконати більш гнучку міграцію реляційної таблиці *country*, зокрема замінити ідентифікатор столиці на вкладений документ, що містить необхідну додаткову інформацію про столицю.

У процесі міграції даних реляційний зв'язок один-до-одного (country.Capital → city.ID) трансформується таким чином: зовнішній ключ country.Capital реляційної таблиці *country* замінюється у документі MongoDB на поле типу Object (вкладений документ), який формується на основі даних таблиці *city* та містить назву столиці і чисельність її населення, отримані за відповідним первинним ключем city.ID.

Додаткові налаштування Mappings (рисунок 6), які треба виконати для поточної міграції, будуть полягати у:

– видаленні запропонованого за замовчуванням поля *Capital*;

– додаванні нового поля *Capital* з реалізацію зв'язка country.Capital → city.ID (тип нового поля Object, джерело даних – стовпець city.Name і city.Population).

У результаті такої міграції в MongoDB створена колекція *collection_country* буде містити в одному документі не лише інформацію про відповідну країну, а й вкладений документ із даними про столицю (її назву та чисельність населення):

```
{
   "_id" : "ABW",
   "Name" : "Aruba",
   "Continent" : "North America",
   "Region" : "Caribbean",
   …
   "Capital" : {
      "Name" : "Oranjestad",
      "Population" : NumberInt(29034)
   }
}
```

У MySQL для отримання даних щодо назви і чисельності населення столиці для конкретної країни необхідно виконати JOIN-запит до двох реляційних таблиць *country* і *city*, тоді як у MongoDB всі необхідні дані доступні безпосередньо в межах одного документа, що спрощує структуру запитів і підвищує ефективність операцій читання.

7. Міграція в MongoDB реляційних таблиць, пов'язаних зв'язком один-до-багатьох

При реалізації міграції реляційної таблиці *country* у MongoDB за замовчуванням кожен документ колекції *collection_country* буде містити лише дані, які безпосередньо зберігалися у таблиці *country*. Інформація про мови, які використовуються у відповідній країні, в реляційній моделі зберігається в окремій таблиці *countrylanguage*.

У Studio 3T є можливість додати до кожного документа колекції *collection_country* інформацію про мови, які використовуються у відповідній країні. Таке додавання можна також реалізувати двома способами.

ISSN 2071-1077(print)
ISSN 2663-2411(online

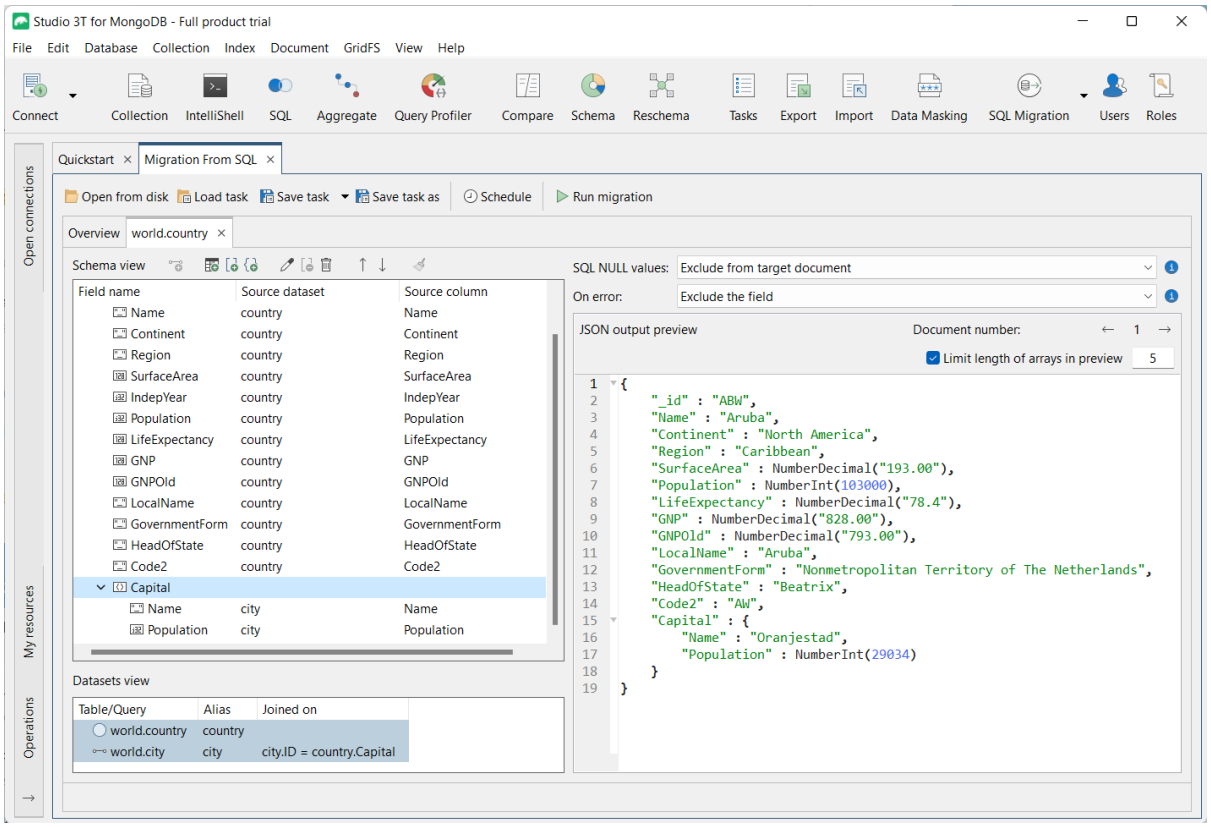

Рис. 6. Попередній перегляд перед міграцією даних

Часткове вбудовування (partial embedding)

Між таблицями *country* і *countrylanguage* існує зв'язок один-до-багатьох: одна країна може використовувати декілька мов. Такий зв'язок реалізується за допомогою зовнішнього ключа countrylanguage.CountryCode, який посилається на первинний ключ country.Code.

У MySQL для отримання інформації про мови, що використовуються в конкретній країні, необхідно виконати JOIN-запит між таблицями *country* та *countrylanguage*.

Однак MongoDB має можливість зберігати значення типу Array і тому за допомогою GUI-клієнта Studio 3T та інструмента Mappings можна налаштувати міграцію зв'язку один-до-багатьох таким чином, що кожен документ колекції *collection_country* буде включати додаткове поле *Languages* типу Array, яке буде містити перелік мов, що використовуються у відповідній країні.

Додаткові налаштування Mappings (рис. 7), які треба виконати для поточної міграції, будуть полягати у додаванні нового поля *Languages* з реалізацію зв'язка countrylanguage.CountryCode → country.Code (тип нового поля Array, джерело даних – стовпець countrylanguage.Language).

У результаті такої міграції в MongoDB створена колекція *collection_country* буде містити в одному документі не лише інформацію про відповідну країну, а й масив *Languages*, який включає назви мов, що використовуються у цій країні:

```
{
  "_id" : "ABW",
  "Name" : "Aruba",
  "Continent" : "North America",
  "Region" : "Caribbean",
  …
```

ISSN 2071-1077(print)
ISSN 2663-2411(online

```
  "Languages" : [
     "Dutch",
     "English",
     "Papiamento",
     "Spanish"
  ]
}
```

На відміну від реляційної бази даних MySQL, де для отримання переліку мов конкретної країни необхідно виконувати JOIN-запит між таблицями *country* та *countrylanguage*, у MongoDB всі необхідні дані доступні безпосередньо в межах одного документа.

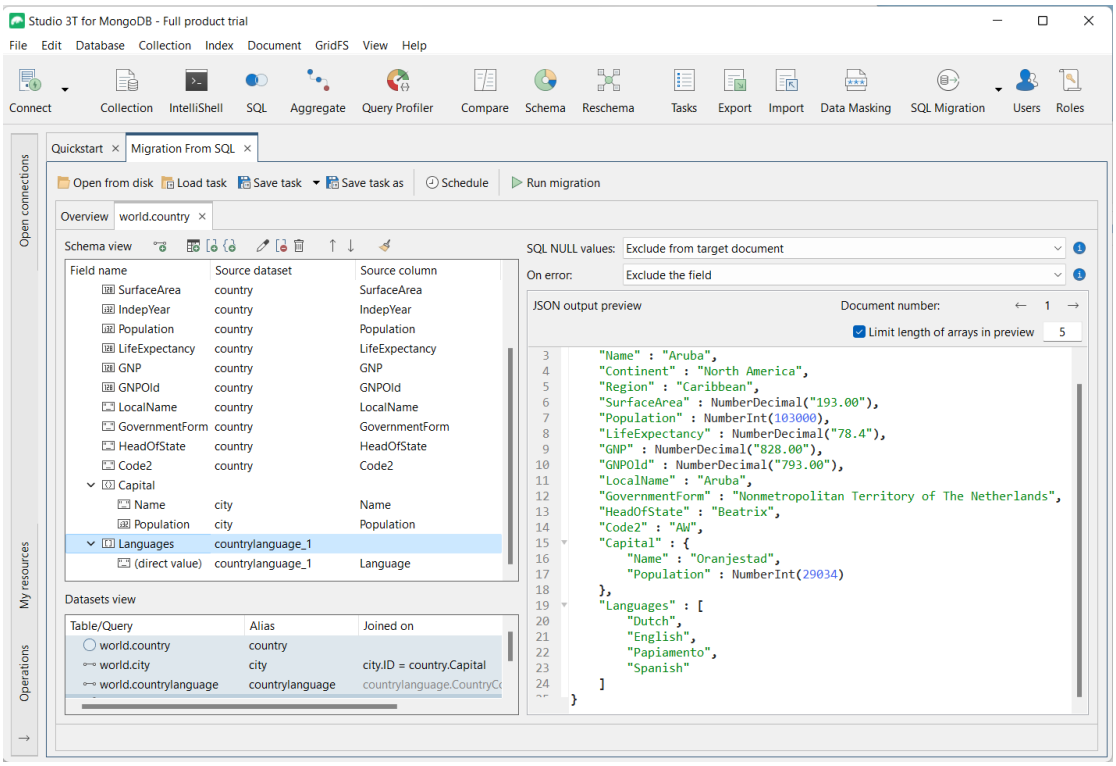

Рис. 7. Попередній перегляд перед міграцією даних

Повне вбудовування (full embedding)

GUI-клієнт Studio 3T за допомогою інструмента Mappings дозволяє виконати більш гнучку міграцію реляційної таблиці *country*, зокрема кожен документ колекції *collection_country* може містити додаткове поле *Languages* типу Array of Objects. Кожен елемент такого масиву буде представляти окрему мову та включати детальну інформацію про неї з таблиці *countrylanguage.*

Додаткові налаштування Mappings (рисунок 8), які треба виконати для поточної міграції, будуть полягати у додаванні нового поля *Languages* з реалізацію зв'язка countrylanguage.CountryCode → country.Code (тип нового поля Array of Object, джерело даних – стовпці countrylanguage.Language, countrylanguage.IsOfficial і countrylanguage.Percentage).

У результаті такої міграції в MongoDB створена колекція *collection_country* буде містити в одному документі не лише інформацію про відповідну країну, а й масив вкладених об'єктів *Languages*, який буде зберігати детальну інформацію про мови, що використовуються у цій країні (Language, IsOfficial, Percentage):

```
{  "_id" : "ABW",
```

ISSN 2071-1077(print)
ISSN 2663-2411(online

```
   "Name" : "Aruba",
     …
   "Languages" : [
      {
         "Language" : "Dutch",
         "IsOfficial" : "T",
         "Percentage" : NumberDecimal("5.3")
      },
      …
      {
         "Language" : "Spanish",
         "IsOfficial" : "F",
         "Percentage" : NumberDecimal("7.4")
      }
   ]
}
```

На відміну від реляційної бази даних MySQL, де для отримання повної інформації про мови конкретної країни необхідно виконувати JOIN-запит між таблицями *country* та *countrylanguage*, у MongoDB всі пов’язані дані доступні безпосередньо в межах одного документа, що спрощує структуру запитів і підвищує ефективність операцій читання.

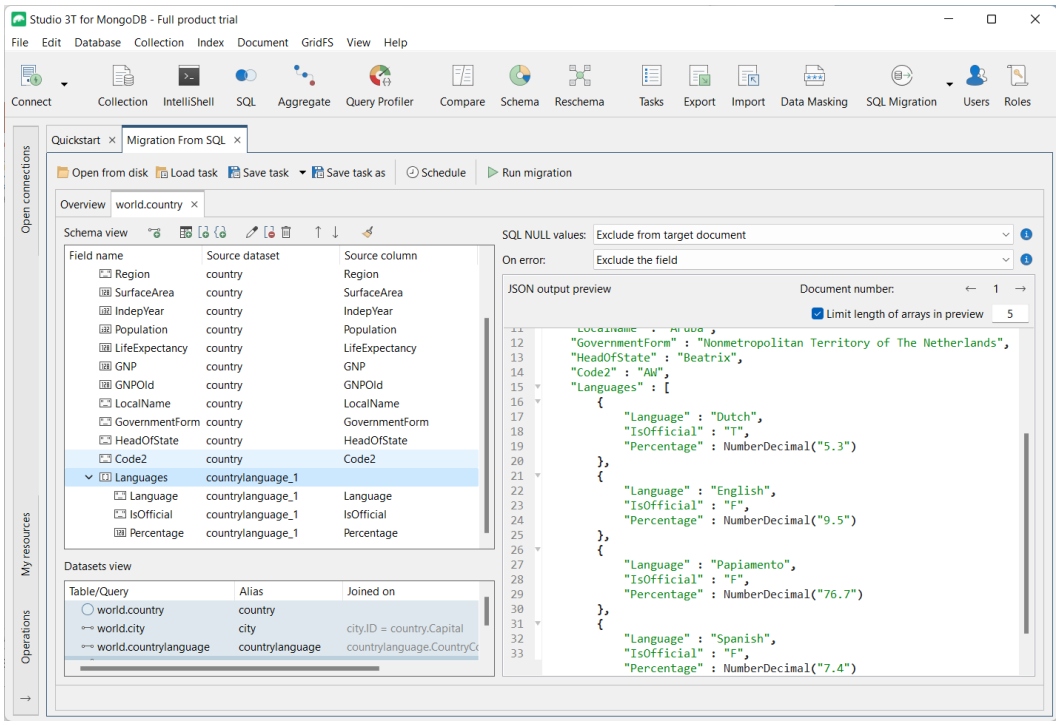

Рис. 8. Попередній перегляд перед міграцією даних

8. Особливості відображення реляційних зв’язків у документо-орієнтованій моделі даних

На основі виконаних міграцій можна виділити ключові особливості перетворення реляційних зв’язків у документо-орієнтовану модель MongoDB. Узагальнені результати аналізу різних стратегій перетворення зв’язків один-до одного і один-до-багатьох наведені у таблицях 4 і 5.

ISSN 2071-1077(print)
ISSN 2663-2411(online

Таблиця 4

Стратегії перетворення реляційного зв’язку один-до-одного в документо-орієнтованій моделі

Стратегія	Опис	Переваги	Недоліки	Випадки застосування
Збереження посилання (referencing)	Зберігання ідентифікатора пов’язаної сутності як скалярного значення	Відсутність дублювання даних; простота оновлення	Потребує виконання окремого запиту або lookup-операції	Пов’язані дані рідко запитуються разом; дані можуть змінюватися незалежно
Часткове вбудовування (partial embedding)	Заміна ідентифікатора на найбільш затребувані поля пов’язаної сутності	Баланс між нормалізацією та денормалізацією; критична інформація доступна безпосередньо	Часткове дублювання даних; потреба синхронізації при оновленні	Необхідний швидкий доступ до ключових атрибутів; повні дані потрібні рідше
Повне вбудовування (full embedding)	Заміна ідентифікатора на вкладений документ з повною інформацією	Оптимальна продуктивність читання; один атомарний запит	Повне дублювання даних; складність синхронізації при оновленні	Дані завжди запитуються разом; рідкі оновлення пов’язаних сутностей

Таблиця 5

Стратегії перетворення реляційного зв’язку один-до-багатьох у документо-орієнтованій моделі

Стратегія	Опис	Переваги	Недоліки	Випадки застосування
Збереження посилань (referencing)	Підлеглі сутності зберігаються в окремій колекції	Відсутність дублювання; підтримка високої кардинальності; незалежне оновлення	Потреба в lookup-операціях; втрата атомарності	Висока кардинальність (сотні-тисячі елементів); часті оновлення підлеглих сутностей
Масив скалярних значень (array of primitives)	Поле типу Array містить список ключових атрибутів підлеглих сутностей	Компактність документів; простота структури; швидкий доступ до базової інформації	Обмежена інформація про підлеглі сутності; неможливість зберігати додаткові атрибути	Необхідна лише базова інформація про підлеглі сутності; обмежена кардинальність (до 100 елементів)
Масив вкладених документів (array of embedded documents)	Поле типу Array містить масив об’єктів з повним набором атрибутів	Повна денормалізація; один атомарний запит; атомарність оновлень на рівні документа	Дублювання даних; обмеження розміру документа (16 МБ); складність синхронізації	Підлеглі дані завжди запитуються разом з головними; обмежена кардинальність (до сотні елементів)

9. Переваги та обмеження використання GUI-клієнта Studio 3T для міграції даних

На основі виконаних міграції даних схеми *world* можна виділити ключові

переваги та обмеження використання GUI-клієнта Studio 3T для перетворення реляційних структур у документо-орієнтовану модель MongoDB.

Studio 3T забезпечує інтуїтивний графічний інтерфейс, який дозволяє налаштовувати міграцію без необхідності написання програмного коду. Інструмент Data Import Wizard надає покрокове керівництво процесом міграції, що значно знижує поріг входу для розробників.

Ключовою перевагою використання Studio 3T є можливість швидкого експериментування з різними підходами до трансформації даних. Розробник може дуже швидко створити кілька різних варіантів міграції однієї таблиці, порівняти результуючі структури документів та обрати оптимальну стратегію.

Інструмент Mapping дозволяє визначати як прості перетворення (заміна зовнішнього ключа на скалярне значення), так і складні (створення вкладених документів або масивів об'єктів). Підтримка типів даних String, Object, Array of String та Array of Object забезпечила в наведених прикладах необхідну гнучкість для реалізації різних стратегій денормалізації.

Schema Explorer дозволяє переглянути структуру результуючих документів ще до фактичного виконання міграції, що критично важливо для верифікації архітектурних рішень та виявлення потенційних проблем на ранніх етапах.

Окрім функцій міграції, Studio 3T надає повноцінне середовище для роботи з MongoDB, включаючи IntelliShell, Visual Query Builder та інструменти для аналізу індексів, що дозволяє не лише виконати міграцію, але й відразу протестувати роботу з новою схемою даних.

Незважаючи на широкі можливості інструменту Mapping, Studio 3T може виявитися недостатнім для реалізації складної бізнес-логіки трансформації даних. Якщо міграція вимагає нетривіальних обчислень або умовної логіки, програмний підхід може бути більш доречним.

Хоча Studio 3T ефективно працює з наборами даних (datasets) середнього розміру, для міграції таблиць з мільйонами записів можуть виникнути проблеми з продуктивністю. Для промислових міграцій великих обсягів даних можуть знадобитися спеціалізовані ETL-платформи.

Використання GUI-інтерфейсу стає обмеженням, коли необхідно інтегрувати міграцію в автоматизовані CI/CD-pipelines. Studio 3T не надає повноцінного API для програмного керування процесом міграції.

Також інструмент не забезпечує вбудованих механізмів для комплексної валідації результатів міграції. Перевірка коректності трансформації потребує додаткових зусиль з боку розробника.

Studio 3T є комерційним продуктом, а отже для використання повного набору функцій міграції необхідна платна ліцензія, що може бути суттєвим фактором для невеликих проєктів або навчальних закладів.

Таким чином, Studio 3T оптимально підходить для:

- прототипування та дослідження різних стратегій міграції на початкових етапах проєктування;
- міграції datasets малого та середнього розміру (до сотень тисяч записів);
- навчальних та дослідницьких проєктів;
- разових міграцій, які не потребують глибокої інтеграції в автоматизовані процеси.

Для промислових міграцій великих обсягів даних або систем зі складною бізнес-логікою доцільно розглянути комбінований підхід: використання Studio 3T для прототипування стратегії міграції та програмної реалізації фінального

рішення.

10. Висновок

У статті досліджено особливості міграції даних з реляційної моделі в документо-орієнтовану на прикладі MongoDB з використанням GUI-клієнта Studio 3T. На основі реляційної схеми *world* СКБД MySQL проаналізовано практичні аспекти трансформації окремих таблиць, а також реляційних зв’язків один-до-одного і один-до-багатьох у відповідні структури документів MongoDB.

У ході дослідження показано, що перехід від реляційної в документо-орієнтованої модель потребує переосмислення підходів до проєктування схеми даних, зокрема відмови від JOIN-операцій на користь денормалізації та вбудовування пов’язаних сутностей. Продемонстровано різні стратегії відображення реляційних зв’язків у MongoDB – збереження посилань, часткове та повне вбудовування, а також використання масивів скалярних значень і масивів вкладених документів. Показано, що вибір конкретної стратегії має ґрунтуватися на характері доступу до даних, частоті оновлень та вимогах до продуктивності.

Практичні експерименти підтвердили, що застосування денормалізованих структур у MongoDB дозволяє зменшити кількість запитів до бази даних і підвищити ефективність операцій читання порівняно з реляційною моделлю, особливо у випадках, коли пов’язані дані часто використовуються разом. Водночас такі підходи супроводжуються дублюванням даних і потребують додаткових механізмів підтримки консистентності.

Оцінка можливостей GUI-клієнта Studio 3T показала, що цей інструмент є ефективним засобом для прототипування та дослідження різних стратегій міграції даних. Візуальний інтерфейс, підтримка mapping’у зв’язків та можливість попереднього перегляду результуючих документів значно спрощують процес аналізу та верифікації архітектурних рішень. Разом із тим, Studio 3T має обмеження щодо продуктивності при роботі з великими обсягами даних та реалізації складної бізнес-логіки трансформації, що зумовлює доцільність використання програмних або ETL-рішень у промислових сценаріях.

Отримані результати можуть бути використані при проєктуванні та реалізації процесів міграції даних з реляційних СКБД у документо-орієнтовані, а також у навчальних і дослідницьких роботах, пов’язаних з вивченням NoSQL-технологій та сучасних підходів до зберігання даних.

Список літератури

1. Reinsel D., Gantz J., Rydning J. Data Age 2025: The Evolution of Data to Life-Critical : IDC White Paper. 2024. URL: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf (дата звернення: 28.01.2026).

2. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, 2017. 616 p.

3. MongoDB Case Study. eBay: Handling massive catalogs with high availability. 2023. URL: https://www.mongodb.com/customers/ebay (дата звернення: 28.01.2026).

4. Studio 3T. SQL to MongoDB Migration: A Comprehensive Guide to Structural Transformation. 2025. URL: https://studio3t.com/knowledge-base/articles/sql-to-

mongodb-migration/ (дата звернення: 28.01.2026).

5. Chodorow K. MongoDB: The Definitive Guide. 3rd Edition. O'Reilly Media, 2019. 514 p.

6. Fowler M., Sadalage P. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley, 2012. 192 p.

7. MongoDB Documentation. Data Modeling Introduction: Embedding vs. Referencing. 2025. URL: https://www.mongodb.com/docs/manual/core/data-modeling-introduction/ (дата звернення: 28.01.2026).

8. Vohra D. ETL Data Migration. Practical Guide to MongoDB. Apress, 2015. P. 285–310.

References

1. Reinsel D., Gantz J., Rydning J. Data Age 2025: The Evolution of Data to Life-Critical : IDC White Paper. 2024. URL: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf (accessed: 28.01.2026).

2. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, 2017. 616 p.

3. MongoDB Case Study. eBay: Handling massive catalogs with high availability. 2023. URL: https://www.mongodb.com/customers/ebay (accessed: 28.01.2026).

4. Studio 3T. SQL to MongoDB Migration: A Comprehensive Guide to Structural Transformation. 2025. URL: https://studio3t.com/knowledge-base/articles/sql-to-mongodb-migration/ (accessed: 28.01.2026).

5. Chodorow K. MongoDB: The Definitive Guide. 3rd Edition. O'Reilly Media, 2019. 514 p.

6. Fowler M., Sadalage P. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley, 2012. 192 p.

7. MongoDB Documentation. Data Modeling Introduction: Embedding vs. Referencing. 2025. URL: https://www.mongodb.com/docs/manual/core/data-modeling-introduction/ (accessed: 28.01.2026).

8. Vohra D. ETL Data Migration. Practical Guide to MongoDB. Apress, 2015. P. 285–310.

Надійшла до редакції 17.02.2026, розглянута на редколегії 18.02.2026

Research on Data Migration Features from Relational to Document-Oriented Model Using Studio 3T GUI Client

The article examines the features of data migration from the MySQL relational database to the MongoDB document-oriented database. The object of the study is the process of migrating a relational data model to a document-oriented one, taking into account the structural differences between SQL and NoSQL approaches. The subject of the study is the methods of representing relational relationships in MongoDB and the tools for their implementation. The purpose of the article is to analyze approaches to migrating tables and one-to-one and one-to-many relationships into the MongoDB document structure using the Studio 3T graphical client. The work uses the relational world schema as an example data source for migration into MongoDB document

collections. Strategies of denormalization, reference preservation, and embedding related entities into documents, as well as the use of arrays of scalar values and arrays of nested objects, are analyzed. It is shown that the choice of data storage model in MongoDB depends on the nature of queries, the frequency of information updates, and system performance requirements. Practical data migration was performed using Studio 3T, and the capabilities of this tool for designing and testing document structure were evaluated. The obtained results confirm the feasibility of using document-oriented databases for systems requiring fast access to related data, and can be used in educational and applied tasks of NoSQL storage design.

Key words: relational databases; document-oriented databases; MongoDB; GUI client; Studio 3T; data migration.

Відомості про авторів:

Шевченко Ілона Володимирівна – канд. техн. наук, доцент кафедри інженерії програмного забезпечення, Національний аерокосмічний університет «Харківський авіаційний інститут». Електронна пошта: ilona.shevchenko@gmail.com, ORCID 0000-0003-0100-0726.

Лучшева Оксана Вадимівна – старший викладач кафедри інженерії програмного забезпечення, Національний аерокосмічний університет «Харківський авіаційний інститут». Електронна пошта: o.luchsheva@khai.edu, ORCID 0000-0003-3855-2815.

Лучшев Павло Олександрович – канд. техн. наук, доцент кафедри інженерії програмного забезпечення, Національний аерокосмічний університет «Харківський авіаційний інститут». Електронна пошта: p.luchshev@khai.edu, ORCID 0000-0002-3522-7622.

About the Authors:

Ilona SHEVCHENKO – Ph.D, associate professor of the software engineering department, National aerospace university «Kharkiv Aviation Institute». Email: ilona.shevchenko@gmail.com, ORCID 0000-0003-0100-0726.

Oksana LUCHSHEVA – senior lecturer of the software engineering department, National aerospace university «Kharkiv Aviation Institute». Email: o.luchsheva@khai.edu, ORCID 0000-0003-3855-2815.

Pavlo LUCHSHEV – Ph.D, associate professor of the software engineering department, National aerospace university «Kharkiv Aviation Institute». Email: p.luchshev@khai.edu, ORCID 0000-0002-3522-7622.