УДК 62-503.5
DOI: 10.25140/2411-5363-2021-1(23)-81-86

Наталія Літвінова, Максим Альперт, Андрій Погульський

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ОБМІНУ ДАНИМИ СУТНОСТЕЙ У РЕЛЯЦІЙНОМУ ПРЕДСТАВЛЕННІ ТА ЇХ ОБРОБКИ

У роботі розглянуто спосіб підвищення ефективності обміну структурованими даними, який може застосовуватися під час розробки програмного забезпечення. Як засоби вирішення проблеми представлено власний формат серіалізації даних, спеціально розроблений з врахуванням специфіки сутностей у реляційній моделі, та механізм програмної обробки даних при десеріалізації, що дозволяє значно підвищити продуктивність роботи застосунку. Фактичним результатом дослідження є побудова концепції експериментального формату та варіант реалізація запропонованого методу обробки даних.

Ключові слова: *обмін даними; програмна обробка даних; формат серіалізації; реляційне представлення.*
Рис.: 1. Бібл.: 7.

Актуальність теми дослідження. Сучасні організації, діяльність яких пов'язана зі сферою товарів та послуг, активно користуються досягненнями технічного розвитку по всьому світу. Сфера розробки програмного забезпечення набула широкого поширення в сучасному житті для всіх напрямків бізнесу. Воно застосовується як для ведення внутрішнього діловодства, так і для просування своєї діяльності на ринку, наприклад, у цілях маркетингу або заради надання зручного способу взаємодії для клієнтів. На сьогодні провідні компанії замовляють інформаційні системи для клієнтів, такі як CRM, програмне забезпечення для управління бізнес-процесами – ERP – та програми на основі робочих процесів.

Якою б не була мета застосунку, будь-яке програмне забезпечення передбачає передачу даних, значна частина з яких зазвичай зберігається в базах даних (БД). З огляду на це забезпечення ефективної передачі та обробки даних є наріжним каменем як для настільних систем, так і для розподілених систем, вебзастосунків та застосунків із мікросервісною архітектурою, які становлять більшу частину ринку програмного забезпечення.

Постановка проблеми. Для вирішення багатьох задач сучасні системи проєктуються не монолітами, а комплексом із декількох різних незалежних частин, які прийнято називати компонентами, або сервісами в мікросервісній архітектурі. До того ж у наш час навіть простим застосункам здебільшого доводиться звертатися до сторонніх сервісів для отримання даних через API. Дуже часто взаємодія між компонентами однієї системи або навіть різними взаємопов'язаними системами відбувається за допомогою мережі, наприклад через HTTP, де дані передаються між сокетами. Такий спосіб взаємодії використовує переважна більшість веб-застосунків.

Як відомо, під час процесу обміну даними головними проблемами з погляду програмістів є: великий обсяг даних, що передаються у пакеті, особливо, якщо йдеться про передачу об'єктів з БД (тут мається на увазі надлишковість), а також можливість швидкої обробки прийнятих даних у програмі для забезпечення найкоротшого часу відгуку на запит користувача. Саме ці аспекти є частиною досліджуваної в межах цієї роботи тематики.

Таким чином, у статті розглянута раніше не висвітлена проблема підвищення ефективності передачі даних сутностей у реляційному представленні та їх обробки з погляду програмної інженерії.

Аналіз останніх досліджень і публікацій. Вивченню технологій забезпечення обміну даними було присвячено багато як вітчизняних [1], так і закордонних публікацій [2; 3], проте в них відсутні дослідження шляхів підвищення ефективності обробки даних. Також серед останніх публікацій у відкритому доступі не знайдено новітніх рішень для вирішення питання зменшення обсягу даних, що передаються між програмними компонентами.

Виділення недосліджених частин загальної проблеми. Незважаючи на численні публікації, в яких вивчаються існуючі формати обміну даними, досі не запропоновано рішення, яке б дозволило покращити обмін даними, які представляють сутності у реляційній моделі. Також недостатньо уваги приділено питанню підвищення ефективності

обробки даних всередині програми. Ця стаття, зокрема, присвячена вирішенню проблеми невідповідності використання існуючих форматів для обміну такими даними через надмірність обсягу пакетів, що передаються через мережу між комп'ютерними процесами, а також можливих шляхів ефективної обробки цих даних всередині програми.

Мета статті. Основною метою роботи є представлення покращеного порівняно з існуючими аналогічними рішеннями формату серіалізації для обміну структурованими даними, зокрема, що мають реляційне представлення. Так, на заміну загальноприйнятих форматів обміну даними (XML, JSON, Protocol Buffers тощо) тут пропонується власний двійковий формат. Також надано опис реалізації методу програмної обробки даних, який дозволяє значно підвищити продуктивність роботи програмного забезпечення.

Виклад основного матеріалу. Наявні в наш час формати серіалізації створювалися як універсальний засіб передачі даних різної структури між програмними компонентами, у мережі, тому вони не є ефективним рішенням для передачі великої кількості об'єктів, що мають спільну структуру. Зважаючи на те, що найпопулярнішим зовнішнім джерелом збереження даних по наш час залишаються реляційні бази даних [4], існує необхідність опрацювати питання підвищення ефективності обміну між програмними компонентами даними, які відповідають реляційній моделі.

Для вирішення описаної проблеми було прийнято рішення створити новий експериментальний формат серіалізації даних, який був би більш пристосований під конкретну обрану модель даних. Так, за своїм призначенням розроблений формат передбачає застосування для реляційних сутностей, представлених у об'єктноорієнтованій парадигмі.

Відомо, що існує два способи представлення серіалізованих даних форматами: текстовий та двійковий (бінарний). Текстовий формат ще прийнято називати human-readable, тобто той, який зрозумілий звичайній людині, а дані, представлені у двійковому форматі, з першого погляду неможливо просто так «прочитати». Очевидно, що «читабельність» текстового формату є його сильною стороною, адже не доведеться писати додаткові інструменти для налагодження введення та виводу, а достатньо просто відкрити серіалізований вивід даних за допомогою текстового редактора та перевірити чи має він правильний вигляд. Проте двійковий формат має інші не менш значущі переваги [5].

Можна виділити такі основні його плюси:

- зазвичай потребує менше циклів процесора;
- може давати значно менші результати, якщо більшість чисел великі або коли нема необхідності кодувати двійкові результати текстово.

У власному форматі було вирішено застосовувати двійкове представлення даних.

Методика вирішення проблеми. Оскільки в цій роботі вирішується спеціалізоване питання — передача даних, що мають реляційне представлення, під час розробки концепції нового формату серіалізації даних ключовими ідеями для мінімізації потоку даних, що передається, стали наступні:

- позбутися дублювання спільної інформації для об'єктів (наприклад, такої, як назви властивостей), яка зазвичай повторюється для кожного об'єкта у відомих форматах;
- розмістити пов'язану, тобто не основну інформацію щодо об'єктів, у кінці повідомлення, так щоби надати можливість використання безпосередньо самого об'єкта одразу, без необхідності очікування, поки завантажаться другорядні пов'язані з ним менш важливі дані;
- при можливості зменшити кількість значень властивостей об'єкта, що передаються, шляхом виключення з потоку «нульових» властивостей, тобто відсутніх даних у конкретному об'єкті.

Звісно, економія обсягу даних, що передаються, здатна показати максимальний результат при умовах серіалізації деякої кількості об'єктів, які мають реляційне представлення, тобто для таких об'єктів, чия структура визначена спільною схемою. За цих умов чим більше об'єктів ми будемо серіалізувати, тим більший виграш може дати використання такого формату, тобто порівняно з існуючими форматами, цей дозволяє зекономити для кожного об'єкта обсяг даних, який буде передаватися.

Реалізація. Предметом використання розробленого формату даних є будь-який програмний продукт, виконаний у об'єктно орієнтованій парадигмі, в якому певній сутності БД співвідноситься відповідний програмний тип даних. Тоді усі об'єкти конкретного типу для передачі даних можуть бути згруповані по відповідним контейнерам – наборам даних, які для зручності далі будемо називати «джерело даних», а кожне джерело даних представляється за допомогою розробленого формату.

Загальна структура поточної версії формату для обміну даними зображена на рис. 1. Як видно, структуру формують сегменти підпису, метаданих та даних.

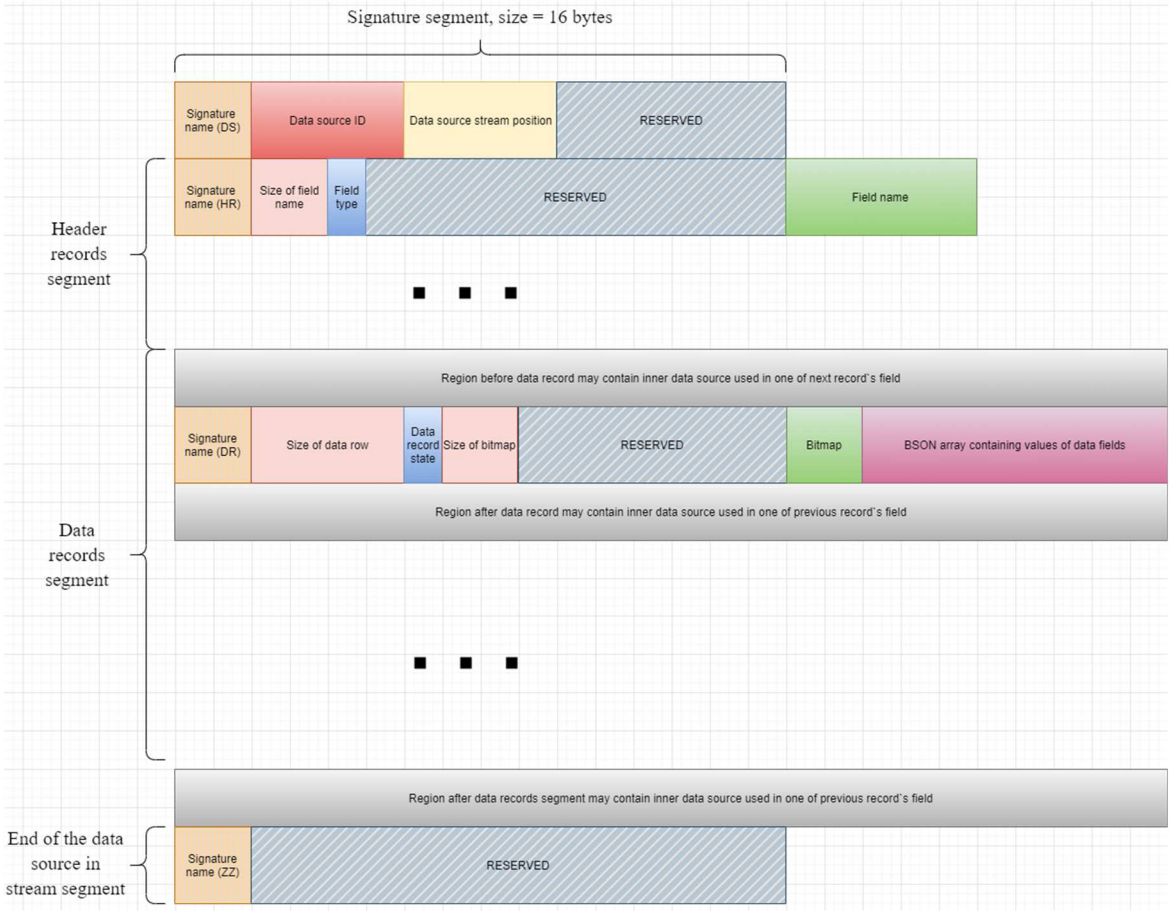

Рис. 1. Структура формату даних

Як видно з рисунку, на початку кожного логічного блока у сегменті обов'язково зазначається його тип (на рисунку він позначений як «Signature type»), що ідентифікує призначення відповідного блока. Так, наразі виділено чотири типи логічно виділених блоків:

- DS (Data Source) означає початок нового джерела з даними, який представляє собою контейнер для сутностей однієї структури (схеми);
- HR (Header Row) — опис структури об'єктів цього джерела;
- DR (Data Row) — значення одного об'єкта;
- ZZ сигналізує про кінець конкретного потоку джерела з даними.

Варто зазначити, що кожне джерело даних зберігає в собі схему відповідної сутності, яка описується за допомогою заголовків, які будемо називати терміном «header record», а також самі дані – значення властивостей об'єктів певного типу, для яких введемо термін «data record».

Маємо, що на відміну від таких популярних форматів, як XML та JSON, де назви властивостей об'єктів повторюються для кожного об'єкта, у цьому форматі опис властивостей подібних сутностей зазначається лише один раз, а безпосередньо самі дані по кожному окремому об'єкту – сегмент data record-ів – слідують один за одним одразу за сегментом header record-ів. Саме це допомагає значно зменшити обсяг даних, а також пам'ять і час передачі пакетів, якщо йдеться про «спілкування» у мережі між різними сервісами. По суті, ці показники є найбільш критичними для передачі даних через HTTP.

На початку потоку, що відповідає джерелу, розміщено сегмент підпис (на рисунку позначений як «Signature segment») — це 16-байтові блоки, що містять опис відповідної одиниці. Залежно від типу блоку тут може розміщуватися різноманітна службова інформація: ідентифікатор, інформація про тип, розмір та інше. Підпис «веде» будь-який блок даних або метаданих, на рисунку він зображений перший згори.

Якщо рухатися вниз по рисунку, далі представлено сегмент header record-ів. Як вже було згадано, він містить опис схеми конкретного типу (сутності); кожен рядок у сегменті метаданих представляє специфікацію поля даних, яке відповідає властивості відповідної реляційної сутності. У підписі header record-а міститься назва поля даних та тип даних, за допомогою якого можна визначити, чи ця властивість представляє собою скалярне (просте) значення, чи є посиланням на інший набір даних (джерело), за ним — власне ім'я описаної в цьому рядку властивості об'єкта.

Після цього, далі у форматі, розміщено сегмент data record-ів, який включає в себе довільну кількість рядків, що містять безпосередньо значення полів, описаних у метаданих, тобто значення властивостей об'єктів обраної сутності. Так, один рядом даних (data record) складається з таких елементів:

у підписі:

- загальний розмір даних об'єкта, тобто розмір bson-масиву, розміщеному в кінці рядку;
- розмір bitmap — спеціальної структури, яка описує присутні в об'єкті властивості зі схеми;

за підписом:

- сам bitmap: один біт відповідає за якесь одне поле, описане у схемі джерела (0 означає відсутність значення, 1 – присутність), дозволяється не писати замикаючі нулі;
- bson-масив, у якому відповідно до зазначено у bitmap порядку слідують значення властивостей.

Біля data record-ів можна спостерігати видовжені сірі блоки. Вони означають, що у відповідних місцях можливі вкладені набори даних інших сутностей, тобто джерела для пов'язаних об'єктів (для нескалярних властивостей об'єкта).

Заштриховані на рисунку блоки сірого кольору з написом «RESERVED» означають, що це місце у підписі блока наразі зарезервоване (визначена кількість байт) і може бути використане для майбутніх потреб.

Якщо вести мову про самі дані об'єктів, то в поточній версії формату вони розташовані у спеціальному JSON-масиві, що має двійковий формат. Цей масив містить значення властивостей об'єкта в строгому порядку, заданому метаданими. З метою зменшення обсягу передачі даних кількість полів даних, описаних у bitmap-структурі, і відповідно кількість розташованих у масиві значень полів, може бути меншою, ніж це заявлено в метаданих, що дозволяє позбутися необхідності передавати порожні або неіснуючі значення.

Іншим засобом для покращення ефективності роботи застосунків є розроблений метод обробки серіалізованих відповідно до даного формату даних. Він надає можливість обробки даних одночасно декількома потоками, що у перспективі здатне значно пришвидшити роботу багатопотокових систем з використанням такого формату. Його реалізація розроблена за допомогою високорівневої мови програмування C#, на платформі .NET Core [6], а в якості джерела даних для тестування було обрано систему керування базами даних PostgreSQL [7]. Для пояснення принципу роботи розробленого методу розглянемо, що відбувається з заповненим відповідно до вищеописаного методу потоком даних, який передається мережею.

Так, дані з потоку зчитуються по логічним блокам та поступово записуються у відповідний об'єкт для взаємодії у коді, назвемо його відповідно до введених раніше термінів Джерело. Джерело містить зв'язаний список, де зберігає всі завантажені в нього дані сутностей. Щойно з потоку було зчитано новий логічний блок з даними та передано Джерелу, Джерело за допомогою System.Threading.Monitor блокує свій список для додання нових даних (Enter). Коли список готовий, Джерело повідомляє всім потокам, які зчитують з нього дані, що можна продовжувати роботу (Pulse) та розблоковує список (Exit). Відповідно усі потоки, які очікували нових даних (Wait), «прокидаються» та здійснюють далі свою роботу.

Для механізму паралельного зчитування було розроблено Курсор. Він повертається Джерелом при ініціюванні читання та використовується для переміщення по даним, запам'ятовує останню позицію у списку даних для кожної задачі-читача. Один курсор може використовуватися декількома потоками, які між собою розділяють задачу читання.

Результати. Представлений у статті формат серіалізації даних для сутностей, що мають реляційне представлення, виділяється серед існуючих провідних аналогів. Згідно з проведеними тестами завдяки використанню описаного формату обсяг даних, що передаються між програмними компонентами, за останніми підрахунками можна зменшити у середньому приблизно до 20-30 % порівняно з існуючими популярними текстовими форматами. Поточна версія реалізації обміну даними за допомогою цього формату була розроблена на платформі .NET Core та спроектована таким чином, що передбачає можливість паралельної обробки даних, які надходять з мережі, що є важливим чинником для пришвидшення швидкодії та підвищення загальної продуктивності роботи програмного забезпечення.

Висновки. У межах цієї статті були запропоновані засоби для підвищення ефективності обміну даними сутностей, що мають реляційне представлення: спеціально розроблений формат серіалізації даних та метод програмної обробки. Це дозволяє підвищити продуктивність програмних продуктів та має позитивний вплив на витрати підприємств, пов'язані з передачею та обробкою даних застосунком, що стосуються як апаратного забезпечення, так і необхідних комп'ютерних потужностей.

Підвищення ефективності обміну даними у програмних системах — значущий крок для покращення економічності та швидкодії під час обміну даними в мережі.

Список використаних джерел

1. Андрущенко Р. Порівняльний аналіз показників ефективності методів серіалізації даних у комп'ютерних мережах. *Технічні науки та технології.* 2019. № 15(1). С. 115-126.

2. Maeda K. Performance evaluation of object serialization libraries in xml, json and binary formats. *Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)*. 2012. Pp. 177-182.

3. Simec A., Maglicic M. Comparison of JSON and XML Data Formats. *Central European Conference on Information and Intelligent Systems*. 2014. Pp. 272-275.

4. 2019 Database Trends – SQL vs. NoSQL, Top Databases. URL: https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use.

5. Marshall C. What's This "Serialization" Thing All About? *C++ FAQ*, 2015. URL: https://web.archive.org/web/20150405013606/http://isocpp.org/wiki/faq/serialization.

6. NET | Free. Cross-platform. Open source. URL: https://dotnet.microsoft.com.

7. PostgreSQL: The world's most advanced open source relational database. URL: https://www.postgresql.org.

References

1. Andrushchenko, R. (2019). Porivnialnyi analiz pokaznykiv efektyvnosti metodiv serializatsii danykh u kompiuternykh merezhakh [Comparative analysis of the performance characteristics of data serialization methods in computer networks]. *Tekhnichni nauky ta tekhnolohii – Technical sciences and technologies,* 15(1), pp. 115-126.

2. Maeda, K. (2012). Performance evaluation of object serialization libraries in xml, json and binary formats. *Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)*, pp. 177-182.

3. Simec, A., Maglicic, M. (2014). Comparison of JSON and XML Data Formats. *Central European Conference on Information and Intelligent Systems*, pp. 272-275.

4. 2019 Database Trends – SQL vs. NoSQL, Top Databases. https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use.

5. Marshall C. What's This "Serialization" Thing All About? *C++ FAQ, 2015*. https://web.archive.org/web/20150405013606/http://isocpp.org/wiki/faq/serialization.

6. .NET | Free. Cross-platform. Open source. https://dotnet.microsoft.com.

7. PostgreSQL: The world's most advanced open source relational database. https://www.postgresql.org.

UDC 62-503.5

Nataliia Litvinova, Maksym Alpert, Andrii Pohulskyi

IMPROVING THE EFFICIENCY OF DATA EXCHANGE OF ENTITIES IN RELATIONAL REPRESENTATION AND THEIR PROCESSING

Nowadays, the area of software development has become widespread among all areas of business, so it is important to ensure high efficiency of applications. Due to the fact that data operating is an integral part of any program, the question how to increase the efficiency of data transmission and processing appeared.

Existing technologies for data exchange are designed primarily to meet the criteria of universality and clarity, but these technologies turn out to be redundant or insufficient for software products operating data in relational representation.

Many domestic and foreign publications have been devoted to the study of data exchange formats, but all lack research on ways to improve the efficiency of software data processing. Also, the public domain doesn't contain any latest publications solving the issue of reducing the amount of data being transmitted between software components. Improving the efficiency of exchange of data in relational representation, using a specially designed data serialization format and method of processing.

The article is devoted to presenting own improved data serialization format for data exchange of entities in relational representation, as well as providing a method of software data processing, which can significantly increase productivity.

This paper reveals the problem of exchanging data in relational representation and the disadvantages of using existing serialization formats for transferring relational objects in software systems. The explanation to the decision proposed here is given with the help of the description of the applied technique for problem solving, the graphic scheme of the data format designed and an example of program realization.

The actual result of the study is the construction of the concept of own experimental data serialization format as a way to partially solve the problem described and a detailed description of its structure. The article also describes the .NET Core implementation of the proposed method of efficient processing of data retrieved as a result of deserialization from the format presented.

The following tools to improve the efficiency of data exchange for entities with a relational representation were proposed in the article: a specially developed data serialization format and the method of its software processing.

Keywords: *data exchange; software data processing; serialization format; relational representation.*

Fig.: 1. References.: 7.

Літвінова Наталія Олександрівна — студентка-магістрантка, Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського" (просп. Перемоги, 37, м. Київ, 03056, Україна).
Litvinova Nataliia — master student, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" (37 Peremohy Av., 03056 Kyiv, Ukraine).
E-mail: natalsha07@gmail.com
ORCID: https://orcid.org/0000-0001-8079-7328

Альперт Максим Іоганович — студент-магістрант, Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського" (просп. Перемоги, 37, м. Київ, 03056, Україна).
Alpert Maksym — master student, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" (37 Peremohy Av., 03056 Kyiv, Ukraine).

Погульський Андрій Миколайович — студент-бакалавр, Коледж інженерії та управління Національного авіаційного університету (вул. Метробудівська, 5-а, м. Київ, 03065, Україна).
Pohulskyi Adrii — bachelor- student, College of Engineering and Management of the National Aviation University (5-a Metrobudivska Str., 03065 Kyiv, Ukraine).

Літвінова Н., Альперт М., Погульський А. Підвищення ефективності обміну даними сутностей у реляційному представленні та їх обробки. *Технічні науки та технології.* 2021. № 1(23). С. 81-86.