Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Технологія трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент 3 курсу, група КНс-20-1* _____ *І.О. Андросюк*
Курс, група виконавця | Підпис | Ініціали, прізвище

Керівник: *к.т.н., доцент кафедри КН* _____ *О.А.Пасічник*
Науковий ступінь, посада | Підпис | Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КН* _____ *Р.О. Багрій*
Науковий ступінь, посада | Підпис | Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор _____ *О.В. Бармак*
Підпис | Ініціали, прізвище
6 06 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*
_____
(підпис)
д.т.н., професор О.В. Бармак
«6» 03 *2023* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Технологія трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів»*

2. Завдання видано студенту *Андросюку Івану Олександровичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *доцент кафедри КН Пасічник Олександр Анатолійович*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «1» 03 2023 р. № 5

5. Дата видачі завдання студенту: «3» 03 202 р.

6. Зміст пояснювальної записки (перелік задач) та вихідні дані. Провести аналіз предметної області, сучасних платформ електронної комерції, сучасних тенденцій покращення ефективності електронної комерції. Виконати огляд принципів побудови запитів. Реалізувати технологію трансферу даних з використанням декомпозиції запитів. Виконати програмну реалізацію технології трансферу даних для з використанням декомпозиції запитів. Провести експериментальне тестування інформаційної технології. Основою реалізації технології є платформа Magento 2 e-commerce.

7. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником	грудень 2022	Виконано
2	Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження	січень 2023	Виконано
3	Робота над розділом 1 – Характеристика предметної області та постановка задачі	січень 2023	Виконано
4	Робота над розділом 2 – Технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів	березень 2023	Виконано
5	Робота над розділом 3 – Програмна реалізація технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів	квітень 2023	Виконано
6	Оформлення пояснювальної записки згідно вимог	травень 2023	Виконано
7	Попередній захист кваліфікаційної роботи бакалавра	травень 2023	Виконано
8	Захист кваліфікаційної роботи бакалавра на засіданні Екзаменаційної комісії	червень 2023	Виконано

Виконавець: студент 3 курсу, група КНс-20-1 _____ І.О. Андросюк
Курс, група виконавця — Підпис — Ініціали, прізвище

Керівник: к.т.н., доцент кафедри КН _____ О.А. Пасічник
Науковий ступінь, посада — Підпис — Ініціали, прізвище

Анотація

Тема кваліфікаційної роботи бакалавра: *Технологія трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КНс-20-1 Андросюк Іван Олександрович*

Керівник кваліфікаційної роботи бакалавра: *к.т.н., доцент кафедри КН Пасічник Олександр Анатолійович*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
61	38	1	25	2

Метою кваліфікаційної роботи бакалавра є розробка технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів. Для досягнення поставленої мети визначені наступні задачі дослідження: провести аналіз предметної області, сучасних платформ електронної комерції, сучасних тенденцій покращення ефективності електронної комерції; виконати огляд принципів побудови запитів; реалізувати технологію трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів; виконати програмну реалізацію технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів; провести експериментальне тестування інформаційної технології.

Результатом виконання кваліфікаційної роботи бакалавра є технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів та її реалізація як інформаційної системи.

Ключові слова: запит, декомпозиція запитів, трансфер даних.

Виконавець: *студент 3 курсу, група КНс-20-1* — *І.О. Андросюк*

Курс, група виконавця — Підпис — Ініціали, прізвище

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
ORM	Object-Relational Mapping
AJAX	Asynchronous Javascript and XML
КРБ	Кваліфікаційна робота бакалавра
КН	Комп’ютерні науки
DB	База даних

Вступ

Кваліфікаційна робота бакалавра присвячена розробці технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів

Актуальність теми. Інтенсивний розвиток, зростання функціональності, розширення сфер застосування та поглиблення інтелектуалізації вирішуваних задач при впровадженні інформаційних систем та технологій є стійкою та головною тенденцією розвитку людства останніми десятиліттями та, певно, залишиться такою в досяжному майбутньому. Однією з сфер діяльності, яка є флагманом у використанні цифрових технологій виступає торгівля. Наявні величезні успіхи та досягнення в цій області висувають все нові виклики для забезпечення стійкого росту галузі. Побічним ефектом цих успіхів та цього зростання, який необхідно подолати, є ефективне керування все зростаючою кількістю замовлень із дотримання прийнятних часових параметрів їх опрацювання. Пропонованими шляхами вирішення цієї задачі є, як використання сучасних платформ, так й заходи щодо покращення структури потоків інформації в межах наявних рішень.

Мета і задачі роботи. Мета кваліфікаційної роботи полягає в створенні технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів.

Для досягнення поставленої мети визначені наступні задачі дослідження:

- провести аналіз предметної області, сучасних платформ електронної комерції, сучасних тенденцій покращення ефективності електронної комерції;
- виконати огляд принципів побудови запитів;
- реалізувати технологію трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів;
- виконати програмну реалізацію технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів;
- провести експериментальне тестування інформаційної технології.

Об’єкт дослідження – процес побудови технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів.

Предмет дослідження – моделі, алгоритми та засоби для створення технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів.

Розділ 1 Характеристика предметної області та постановка задачі

1.1 Аналіз предметної області

Magento 2 — це популярна платформа електронної комерції, яка надає надійний набір функцій для створення та керування онлайн-магазинами. Ось деякі з ключових функцій і переваг Magento 2 як платформи електронної комерції, що пропонує потужну основу для створення та керування онлайн-магазином, надаючи змогу створити багатофункціональний, масштабований та персоналізований досвід електронної комерції:

− Гнучкий і масштабований: Magento 2 дуже гнучкий і може бути налаштований відповідно до конкретних потреб вашого бізнесу. Ця платформа може працювати з малими та великими онлайн-магазинами та підходить як для електронної комерції B2C (бізнес-споживач), так і для B2B (бізнес-бізнес).

− Багатий набір функцій: Magento 2 пропонує великий та різноманітний набір функцій для вдосконалення вашого онлайн-магазину. Включає керування каталогом, керування замовленнями, керування клієнтами, інтеграцію платіжного шлюзу, варіанти доставки, маркетингові інструменти тощо. Ці функції дають змогу створювати безперебійний досвід покупок для клієнтів.

− Адаптивний дизайн: Magento 2 підтримує адаптивний веб-дизайн, що означає, що онлайн-магазин автоматично налаштує свій макет і зовнішній вигляд, щоб забезпечити оптимальний досвід перегляду на різних пристроях, таких як настільні ПК, планшети та смартфони.

− Розширення та теми: Magento Marketplace пропонує велику колекцію розширень і тем, які дозволяють розширити функціональність і налаштувати зовнішній вигляд магазину. За допомогою Marketplace можна знайти різні безкоштовні та платні розширення для покращення різних аспектів онлайн-бізнесу.

− Можливість кількох магазинів: за допомогою Magento 2 можна керувати кількома онлайн-магазинами з однієї панелі адміністратора. Ця функція особливо

корисна, якщо у вас є кілька брендів, продуктових ліній або регіональних магазинів, які потребують різних конфігурацій.

− Продуктивність і безпека: Magento 2 розроблено для забезпечення високої продуктивності та масштабованості. Платформа включає механізми кешування, оптимізацію бази даних та інші покращення продуктивності, щоб забезпечити швидке завантаження сторінок і плавний перегляд для потреб клієнтів. Крім того, Magento приділяє увагу безпеці та регулярно випускає виправлення та оновлення для усунення вразливостей.

− Підтримка спільноти: Magento має велику й активну спільноту розробників, дизайнерів і користувачів, які надають підтримку, діляться знаннями та роблять внесок у вдосконалення платформи. Є багато ресурсів, форумів та документацій, які можуть допомогти із будь-якими питаннями чи проблемами, з якими може зіткнутися як розробник, так і користувач [1].

Magento 2 має кілька переваг, які роблять його кращим вибором для багатьох компаній порівняно з іншими платформами електронної комерції. Ось кілька причин, чому багато хто вважає Magento 2 кращим:

− Масштабованість і продуктивність: Magento 2 створено для обробки великого трафіку та великомасштабних операцій електронної комерції. Архітектура та механізми кешування дозволяють швидше завантажувати сторінки, покращувати продуктивність і масштабованість. Це робить платформу придатною для підприємств із зростаючими або складними онлайн-магазинами.

− Гнучкість і налаштування: Magento 2 пропонує широкі можливості налаштування, що дозволяє компаніям пристосовувати свої онлайн-магазини до своїх конкретних потреб. Можливості платформи забезпечують модульну архітектуру, яка дозволяє розробникам створювати власні функції, розширення та інтеграції. Гнучкість платформи робить її адаптованою до різних галузей і бізнес-моделей.

− Можливості B2B: Magento 2 має надійні функції, розроблені спеціально для електронної комерції B2B. Він пропонує розширені параметри ціноутворення та пропозиції, спеціальні каталоги, керування обліковим записом компанії та

інтеграцію з системами ERP (система автоматизації бізнес-процесів підприємства). Ці можливості роблять його придатним вибором для підприємств, які обслуговують оптових або бізнес-клієнтів.

– Екосистема розширень і тем: Magento 2 має процвітаючий ринок, який пропонує широкий спектр розширень і тем. Ці розширення та теми дозволяють компаніям розширити функціональні можливості платформи, покращити взаємодію з користувачем і налаштувати дизайн своїх онлайн-магазинів. Наявність різноманітної екосистеми надає підприємствам можливості для задоволення їхніх конкретних вимог.

Хоча Magento 2 пропонує численні переваги, під час оцінки платформ електронної комерції важливо враховувати конкретні бізнес-потреби, бюджет і технічний досвід. Різні платформи можуть досягти успіху в певних сферах або задовольняти певні галузі [1].

1.2 Аналіз сучасних платформ електронної комерції

Якщо говорити про «аналогову платформу» в контексті електронної комерції Magento 2, то не існує конкретної платформи, яку можна було б розглядати як точний аналог або еквівалент. Однак існують альтернативні платформи електронної комерції, які виконують подібні цілі і пропонують функції, порівняні з Magento 2. Список декількох помітних аналогових платформ:

− WooCommerce — це популярна платформа електронної комерції, яка функціонує як плагін для WordPress, широко використовуваної системи керування вмістом. Забезпечує повну інтеграцію з веб-сайтами WordPress, дозволяючи користувачам перетворювати свої сайти WordPress на повнофункціональні онлайн-магазини. WooCommerce пропонує функції для керування продуктами, обробки замовлень, платіжних шлюзів, а також різноманітні розширення та теми для налаштування. Особливо підходить для малого та середнього бізнесу або осіб, які вже знайомі з WordPress [4].

− Shopify — це комплексна платформа електронної комерції, яка надає комплексне рішення для створення та керування онлайн-магазинами. Пропонує інтуїтивно зрозумілий інтерфейс, попередньо розроблені шаблони та широкий спектр функцій, таких як керування продуктами, безпечні платіжні шлюзи, відстеження запасів, виконання замовлень, маркетингові інструменти тощо. Shopify відомий своїм зручним підходом, що робить його доступним для початківців і тих, хто не має великих технічних знань [2].

− BigCommerce — це хмарна платформа електронної комерції, яка надає повністю розміщене рішення для створення онлайн-магазинів і керування ними. Платформа пропонує функції для керування каталогом продуктів, безпечні платіжні шлюзи, обробку замовлень, маркетингові інструменти та інтеграцію з популярними сторонніми службами. BigCommerce зосереджується на забезпеченні масштабованості, надійності та широкому діапазоні параметрів налаштування. Як правило обслуговує підприємства різного розміру, від малих до корпоративних [3].

− PrestaShop — це платформа електронної комерції з відкритим кодом, яка дозволяє користувачам створювати онлайн-магазини та керувати ними. Надає ряд функцій, таких як керування продуктами, обробка замовлень, численні платіжні шлюзи, маркетингові інструменти, а також різноманітні теми та модулі для налаштування. PrestaShop пропонує гнучкість і можливості налаштування для користувачів, які віддають перевагу рішенню з відкритим кодом [5].

Кожна з цих аналогових платформ має свої сильні сторони, особливості та цільову аудиторію. Вони пропонують різні рівні налаштування, масштабованості та простоти використання. Вибираючи аналогову платформу, важливо враховувати конкретні бізнес-потреби, бюджет, технічну експертизу та плани розвитку. Серед найпопулярніших можна виділити три наступні платформи: Shopify, Woocommerce, BigCommerce. Проте лідером серед великого бізнесу залишається платформа Magento 2, яка має низку переваг серед аналогових платформ.

Magento 2 та Shopify:

− Архітектура: Magento 2 — це самостійна платформа, яка вимагає серверної інфраструктури та технічних знань для налаштування та обслуговування. Shopify, з іншого боку, є повністю розміщеним рішенням, тобто воно піклується про хостинг, безпеку та технічні аспекти роботи онлайн-магазину.

− Налаштування: Magento 2 надає широкі можливості налаштування, дозволяючи розробникам змінювати основні функції та створювати власні розширення. Shopify пропонує налаштування за допомогою системи тем, API та інтеграції додатків. Однак він може мати обмеження порівняно з рівнем налаштування, доступним у Magento 2.

− Масштабованість: як Magento 2, так і Shopify можуть обробляти масштабовані операції електронної комерції. Архітектура Magento 2 створена для обробки великих обсягів трафіку, складних каталогів і великих баз даних продуктів. Shopify надає масштабовану інфраструктуру та може обслуговувати магазини з високим трафіком, але Shopify може мати певні обмеження порівняно з Magento 2 щодо складних або коплексних налаштувань.

− Продуктивність: Magento 2 пропонує функції оптимізації продуктивності та оптимізацію на рівні сервера, що може сприяти підвищенню продуктивності. Shopify надає оптимізоване середовище хостингу, забезпечуючи хорошу продуктивність більшості магазинів. Однак рівень оптимізації продуктивності може відрізнятися залежно від конкретного налаштування та конфігурацій.

− Magento 2 та Wootomerce:

− Архітектура: Magento 2 має надійну та масштабовану архітектуру, яка підтримує великі каталоги продуктів, численні магазини та складні бізнес-вимоги. Він використовує стек LAMP (Linux, Apache, MySQL, PHP) і дотримується архітектури MVC (Model-View-Controller). З іншого боку, WooCommerce створено як плагін WordPress і відповідає архітектурі WordPress.

− Налаштування: Magento 2 пропонує широкі можливості налаштування, що дозволяє розробникам змінювати основну функціональність, створювати власні модулі та змінювати шаблони. WooCommerce надає параметри

налаштування переважно за допомогою плагінів і тем, які можна розширити за допомогою хуків і фільтрів у WordPress.

− Масштабованість: Magento 2 відомий своєю масштабованістю та здатністю обробляти великомасштабні операції електронної комерції. Magento 2 платформа підтримує розширені механізми кешування, оптимізацію бази даних і розподілену інфраструктуру. WooCommerce підходить для малих і середніх підприємств і може вимагати додаткової оптимізації для обробки високого рівня обсяги перевезень і комплексні каталоги.

− Продуктивність: Magento 2 пропонує функції оптимізації продуктивності, такі як кешування повної сторінки, підтримка Varnish і оптимізація бази даних, які сприяють швидшому завантаженню сторінок. Продуктивність WooCommerce може залежати від таких факторів, як середовище хостингу, встановлені плагіни та практики оптимізації веб-сайту.

Magento 2 та BigCommerce:

− Архітектура: і Magento 2, і BigCommerce є повністю розміщеними платформами електронної комерції. Magento 2 вимагає серверної інфраструктури та технічних знань для налаштування та обслуговування, а BigCommerce піклується про хостинг, безпеку та технічні аспекти.

− Налаштування: Magento 2 надає широкі можливості налаштування, дозволяючи розробникам змінювати основні функції та створювати власні розширення. BigCommerce пропонує налаштування за допомогою своєї системи тем, API та інтеграції програм. Однак Magento 2 забезпечує вищий рівень гнучкості та можливостей налаштування порівняно з BigCommerce.

− Масштабованість: архітектура Magento 2 розроблена для обробки великомасштабних і складних операцій електронної комерції, що робить її високо масштабованою. BigCommerce також розроблений для масштабування та забезпечує інфраструктуру, яка може обслуговувати великий трафік і зростаючі магазини.

− Продуктивність: і Magento 2, і BigCommerce мають функції оптимізації продуктивності та пропонують хостингове середовище, оптимізоване для

електронної комерції. У Magento 2 продуктивність можна додатково підвищити за допомогою оптимізації на рівні сервера та механізмів кешування. BigCommerce надає оптимізоване середовище хостингу, забезпечуючи хорошу продуктивність більшості магазинів.

1.3 Аналіз сучасних тенденцій покращення ефективності електронної комерції

Існує кілька методів підвищення ефективності та продуктивності Magento 2, їх можна розділити на такі категорії: оптимізація кешування, оптимізація продуктивності бази даних, оптимізація коду та конфігурації, конфігурація та оптимізація сервера, розширення та оптимізація модулів. Magento 2 Optimization описує наступні методи [6]:

Оптимізація кешування:

− Увімкнене кешування повної сторінки: Magento 2 пропонує вбудоване кешування повної сторінки, щоб зберігати відтворені сторінки та швидко показувати їх відвідувачам. Увімкнення кешування повної сторінки значно покращує продуктивність. Ви можете налаштувати його в панелі адміністратора Magento.

− Впровадження Varnish Cache: Varnish — це потужний прискорювач HTTP, який може кешувати та обслуговувати сторінки до того, як вони досягнуть Magento. Завдяки інтеграції Varnish із Magento 2 можна розвантажити сервер і покращити час відповіді для статичного вмісту.

− Використання Redis або Memcached: Magento 2 підтримує Redis і Memcached як механізми кешування. Налаштування та використання цих технологій кешування може підвищити продуктивність за рахунок кешування часто використовуваних даних, таких як конфігурація, запити до бази даних і зберігання сеансів.

Оптимізація продуктивності бази даних:

− Індексування: Magento 2 має вбудований індексатор, який оптимізує запити до бази даних. Правильне налаштування та оновлення всіх необхідних індексів, забезпечує підвищення швидкісті роботи з базою даних.

− Очищення бази даних: регулярне очищуння непотрібнмих даних з бази даних, такі як журнали, старі замовлення та покинуті візки. Це допомагає оптимізувати продуктивність бази даних і підвищити загальну ефективність.

Оптимізація коду та конфігурації:

− Увімкнений режим виробництва: Magento 2 надає різні режими для розробки та виробництва. Перехід у робочий режим вимикає функції розробника та покращує продуктивність за допомогою різноманітних оптимізацій.

− Зменення та об’єднання JavaScript і CSS: Magento 2 має вбудовані інструменти для мінімізації та об’єднання файлів JavaScript і CSS. Об’єднання та стиснення цих ресурсів зменшує кількість HTTP-запитів і покращує час завантаження сторінки.

− Оптимізація зображень: великі розміри файлів зображень можуть значно вплинути на час завантаження сторінки. Стискання та процес оптимізації зображення, зменшує їх розмір файлу без шкоди для якості. Для Magento 2 доступні кілька інструментів і розширень для оптимізації зображення.

Конфігурація та оптимізація сервера:

− Використання швидкого веб-серверу: продумане використання веб-серверів, таких як Nginx або Apache, із відповідними конфігураціями для кращої продуктивності. Ці сервери можуть ефективно обробляти одночасні запити.

− Балансування навантаження: якщо магазин отримує високий трафік, реалізація балансування навантаження на кількох серверах може розподілити навантаження та покращити час відповіді.

− Кешування на рівні сервера: реалізація механізму кешування на рівні сервера, такі як кешування коду операції (наприклад, PHP OPcache), щоб кешувати скомпільований код PHP і покращувати продуктивність виконання PHP.

Розширення та оптимізація модулів:

− Аудит розширень: регулярно моніторинг та оптимізація розширення сторонніх розробників. Відключення або вимкнення будь-яких невикористовуваних або ресурсомістких розширень, які можуть вплинути на продуктивність.

− Оптимізація коду: оптимізація коду, для забезпечення ефективних запитів до бази даних, мінімізування непотрібніх операцій введення/виведення файлів і дотримання найкращих практик кодування та патернів в Magento 2.

Це лише деякі з методів підвищення ефективності Magento 2. Важливо проаналізувати та оптимізувати конфігурацію, інфраструктуру та налаштування конкретного магазину на основі результатів тестування продуктивності та моніторингу. Регулярний моніторинг і профілювання продуктивності магазину допоможе виявити вузькі місця та області, які потребують оптимізації.

1.4 Огляд принципів побудови запитів

Під час створення запитів до бази даних у Magento 2 існує кілька загальних підходів і технік, які можна використовувати. Серед них можна виділити основні запити до бази даних у Magento 2 [1]:

− Models and Resource Models – Magento 2 використовує моделі та моделі ресурсів для взаємодії з базою даних. Моделі представляють окремі сутності (наприклад, продукти, клієнти), тоді як моделі ресурсів обробляють операції бази даних для цих сутностей. Щоб виконувати операції з базою даних, потрібно використовувати відповідну модель ресурсів для сутності, з якою відбувається взаємодіяю.

− ORM (Object-Relational Mapping) – ORM Magento 2 дозволяє взаємодіяти з базою даних за допомогою об’єктів PHP, абстрагуючи базові запити SQL. Моделі Magento представляють сутності, а колекції представляють набори сутностей. Щоб отримати дані, створюється екземпляр моделі або колекції, налаштувати потрібні фільтри та сортування, а потім викликати такі методи, як load(), save(), delete().

− Query Builder – Magento 2 надає клас Query Builder (Magento\Framework\DB\Select), який дозволяє вам створювати складні SQL-запити за допомогою вільного інтерфейсу. Щоб створити запит, створюється екземпляр Query Builder і ланцюгові методи, такі як from(), join(), where(), groupBy(), orderBy() тощо. Ці методи визначають таблиці, умови, об’єднання та інші умови запиту.

− Direct SQL Queries – Magento 2 дозволяє виконувати необроблені запити SQL безпосередньо за допомогою адаптера підключення до бази даних. Щоб виконати необроблений SQL-запит, потрібно отримати адаптер підключення до бази даних і використати його методи, такі як query(), fetchAll(), fetchRow(), fetchOne() тощо.

Ці абстракції обробляють різні аспекти, такі як безпека, сумісність і зручність обслуговування, і забезпечують більш послідовний і безпечний спосіб взаємодії з базою даних у Magento 2.

1.5 Мета, задачі та вимоги до реалізації інформаційної системи

Мета кваліфікаційної роботи полягає в створенні технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів. Для досягнення поставленої мети визначені наступні задачі дослідження:

- провести аналіз предметної області, сучасних платформ електронної комерції, сучасних тенденцій покращення ефективності електронної комерції;
- виконати огляд принципів побудови запитів;
- реалізувати технологію трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів;
- виконати програмну реалізацію технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів;
- провести експериментальне тестування інформаційної технології.

Розділ 2 Технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів

2.1 Загальні принципи побудови технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів

Створення технології передачі даних для платформи електронної комерції Magento 2 передбачає ефективну обробку запитів даних.

Технологія трансферу даних для платформи Magento 2 e-commerce, грунтується на дотриманні таких відповідних принципів:

− Аналітичність щодо запитів – аналіз вхідніх запитів, щоб зрозуміти їх структуру, параметри та конкретні необхідні дані. Цей аналіз допомагає ефективно декомпонувати запит.

− Декомпозиція запиту – розбиття вхідних запитів на менші підзапити, націлені на певні сутності даних або таблиці. Така декомпозиція допомагає розподілити робоче навантаження та оптимізувати пошук даних.

− Паралельне виконання – використання розкладених підзапитів паралельно, ефективно використовуючи наявні ресурси. Цей підхід скорочує загальний час відповіді на запит за рахунок одночасного отримання даних.

− Кешування – реалізація механізму кешування для зберігання часто використовуваних або статичних даних. Кешування зменшує потребу в повторних запитах до бази даних, підвищуючи продуктивність і зменшуючи навантаження на сервер бази даних.

− Агрегація даних – після виконання підзапитів об’єднаннях отриманих даних в одну відповідь. Цей крок передбачає об’єднання та структурування даних відповідно до вимог початкового запиту.

− Стиснення даних – імплементація методів стиснення даних, щоб мінімізувати розмір переданих даних. Стиснення допомагає зменшити використання пропускної здатності мережі та покращує загальну продуктивність.

− Обробка помилок – застосування надійніх механізмів обробки помилок для обробки винятків або збоїв, які можуть виникнути під час процесу передачі

даних. Належне звітування про помилки та журналювання допомагають ефективно виявляти та вирішувати проблеми.

− Заходи безпеки – використання відповідних заходів безпеки, такі як автентифікація та авторизація, щоб захистити конфіденційні дані під час процесу передачі. Це гарантує, що лише уповноважені особи можуть отримати доступ до даних і маніпулювати ними.

− Оптимізація та моніторинг продуктивності – відстеження продуктивності технології передачі даних і визначчення області для оптимізації. Це може включати аналіз часу виконання запиту, виявлення вузьких місць і оптимізацію схеми бази даних або стратегії індексування.

Платформа Magento 2 e-commerce використовує дав основних типи трансферу даних – імпорт та експорт. Схема процесу імпорту представлена на рисунку 2.1, схема процесу імпорту представлена на рисунку 2.2.

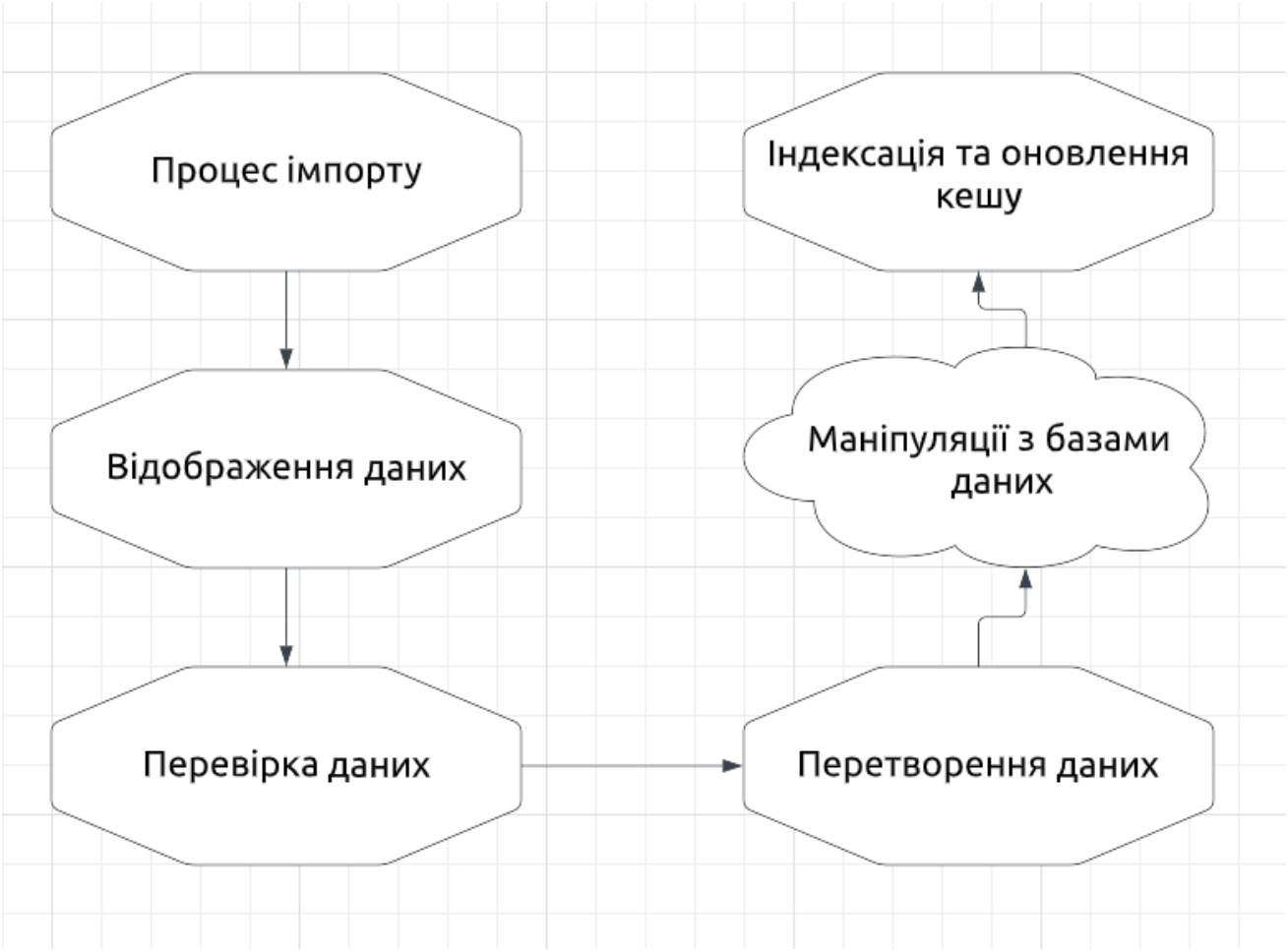


Рисунок 2.1 – Схема процесу імпорту

Процес імпорту – процес імпорту починається із запуску дії імпорту, яку зазвичай ініціює адміністратор, або запланованого завдання. Це запускає потік імпорту даних.

Відображення даних (Data Mapping) – імпортовані дані часто потрібно зіставляти з відповідними полями та атрибутами в структурі бази даних Magento. Цей крок передбачає відображення елементів даних із джерела імпорту у відповідні поля в базі даних.

Перевірка даних – імпортовані дані проходять перевірку, щоб забезпечити їх цілісність і відповідність визначеним правилам і обмеженням. Цей крок перевіряє, чи дані відформатовано правильно, містять дійсні значення та відповідають усім вимогам.

Перетворення даних – на цьому кроці імпортовані дані трансформуються та нормалізуються відповідно до очікуваного формату та структури бази даних Magento. Перетворення даних може включати перетворення типів даних, нормалізацію значень або виконання будь-яких необхідних маніпуляцій з даними.

Маніпуляції з базою даних – після перевірки та трансформації дані вставляються, оновлюються або змінюються в базі даних Magento. Цей крок передбачає виконання операцій з базою даних, таких як вставка нових записів, оновлення існуючих записів або видалення застарілих даних.

Індексація та оновлення кешу – після маніпуляції з базою даних Magento запускає процеси індексації, щоб переконатися, що зміни даних відображаються в пошукових індексах та інших відповідних індексах. Крім того, виконується оновлення кешу, щоб оновити кешовані дані та забезпечити надання точної та актуальної інформації.

Схема процесу імпорту представлена на рисунку 2.2.

Процес експорту – процес експорту починається із запуску дії експорту, яку зазвичай ініціює адміністратор, або запланованого завдання. Це запускає потік експорту даних у Magento.

Вибір даних – на цьому кроці відбувається вибір дані, які потрібно експортувати з бази даних Magento. Це може включати різні сутності, такі як

продукти, категорії, клієнти, замовлення або будь-які інші спеціальні дані.

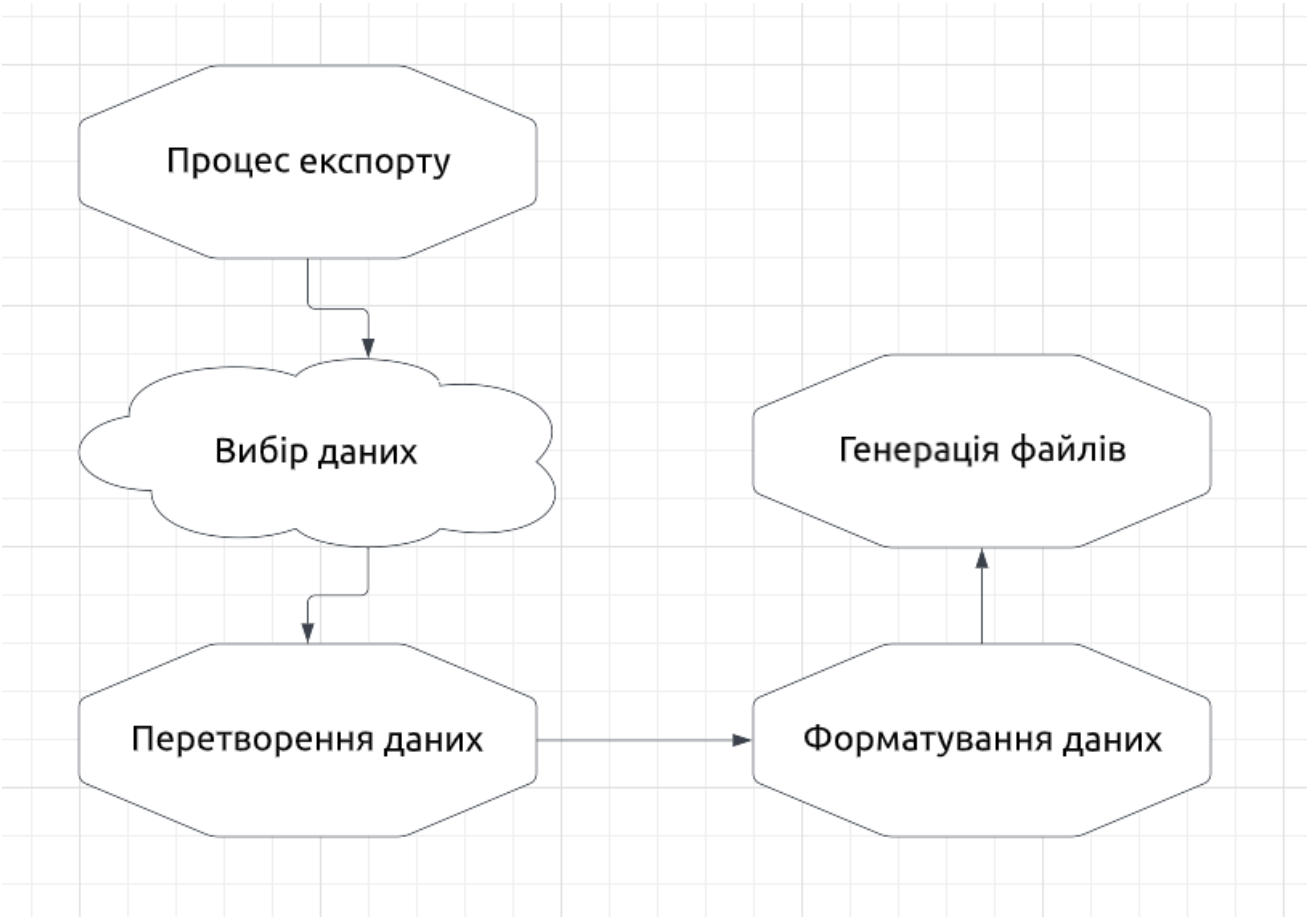


Рисунок 2.2 – Схема процесу експорту

Перетворення даних – вибрані дані можуть бути перетворені відповідно до певних вимог або форматів експорту. Цей крок передбачає перетворення даних у формат, придатний для експорту, наприклад перетворення типів даних, коригування значень або застосування певних бізнес-правил.

Форматування даних – після того, як дані перетворено, вони проходять через форматування, щоб узгодити з бажаним форматом експорту. Цей крок передбачає структурування даних, застосування роздільників, додавання колонтитулів або будь-які інші необхідні коригування форматування.

Генерація файлів – після форматування дані готові до генерації файлу. Відформатовані дані записуються у файл у потрібному форматі експорту, наприклад CSV, XML, Excel або користувальницькі формати. Згенерований файл

можна зберегти локально або надіслати у визначене місце, наприклад на FTP-сервер або службу хмарного зберігання для подальшої роботи з ним.

Схема процесу використання декомпозиції запиту представлена на рисунку 2.3.

Рисунок 2.3 – Схема процесу використання декомпозиції запиту

Запит даних – це початковий запит до бази даних Magento 2 для отримання певних даних, таких як продукти, клієнти або замовлення.

Декомпозиція запиту – на цьому етапі початковий запит розбивається на менші підзапити, які націлені на конкретні сутності даних або таблиці бази даних у Magento 2. Ця декомпозиція допомагає оптимізувати процес пошуку даних.

Підзапити – розбитий запит ділиться на кілька підзапитів, кожен з яких націлений на певний аспект даних у Magento 2. Наприклад, підзапити можуть зосереджуватися на отриманні деталей продукту, інформації про клієнта або даних про замовлення.

Виконати підзапити – підзапити виконуються паралельно або послідовно, залежно від реалізації. Кожен підзапит отримує відповідні дані з бази даних Magento 2 на основі своєї конкретної мети.

Агрегація даних – після виконання підзапитів отримані дані агрегуються в уніфікований результат. Цей крок поєднує та структурує дані відповідно до вимог вихідного запиту.

Остаточний результат запиту – кінцевий результат запиту представляє консолідовані дані, отримані в результаті виконання підзапитів і агрегування результатів. Він представляє дані, які відповідають вимогам початкового запиту до Magento 2.

У Magento 2 існують різні типи об’єктів, які можна імпортувати. Деякі з типів сутностей, які зазвичай імпортуються, включають:

− Продукти – продукти є одними з найбільш часто імпортованих об’єктів у Magento 2. Це включає імпорт атрибутів продукту, зображень, кількості в наявності, цін та інших пов’язаних даних про продукт.

− Категорії – категорії визначають організаційну структуру продуктів у Magento 2. Категорії можна імпортувати, щоб створити або оновити ієрархію категорій у каталозі.

− Клієнти – клієнтів можна імпортувати для заповнення бази даних клієнтів у Magento 2. Це включає інформацію про клієнта, таку як ім’я, адреса електронної пошти, адреса та інші деталі.

− Замовлення – замовлення представляють покупки клієнтів і є важливими для управління продажами та виконанням замовлень. Імпорт замовлень може містити деталі клієнта, інформацію про продукт, статус замовлення та інші пов’язані дані.

− Запаси та інвентар – дані запасів можна імпортувати, щоб оновити кількість продуктів, стан запасів, інформація пов'язана з інвентаризацією.

− Атрибути та набори атрибутів – імпорт атрибутів і наборів атрибутів дозволяє визначати властивості атрибутів продукту та призначати їх різним продуктам.

− Відгуки про продукт – відгуки та рейтинги продуктів можна імпортувати, щоб заповнити розділ оглядів вашого магазину Magento 2.

− Перенаправлення URL – перенаправлення URL-адрес можна імпортувати, щоб керувати перенаправленнями зі старих URL-адрес на нові, зберігаючи рейтинг SEO та забезпечуючи плавний перехід.

Згальна схема процесу трансферу даних для сутностей представлена на рисунку 2.4.

Джерело даних – це джерело, з якого імпортуються дані клієнта, файл CSV, зовнішня база даних або API.

Підготовка даних – на цьому етапі імпортовані дані готуються до обробки. Це може передбачати очищення, форматування та структурування даних для забезпечення узгодженості та сумісності з системою Magento.

Перевірка даних – підготовлені дані перевіряються, щоб гарантувати їх точність, повноту та відповідність попередньо визначеним правилам і обмеженням. Цей крок допомагає виявити та усунути будь-які помилки чи невідповідності в даних.

Перетворення даних – якщо необхідно, підтверджені дані перетворюються відповідно до структури та атрибутів даних Magento. Це включає зіставлення полів, перетворення типів даних і обробку будь-яких необхідних коригувань.

Імпорт сутності – трансформовані дані використовуються для створення нових сутностей або оновлення існуючих у системі Magento. Цей крок передбачає

заповнення атрибутів сутностей, налаштування додаткової інформації про сутність.

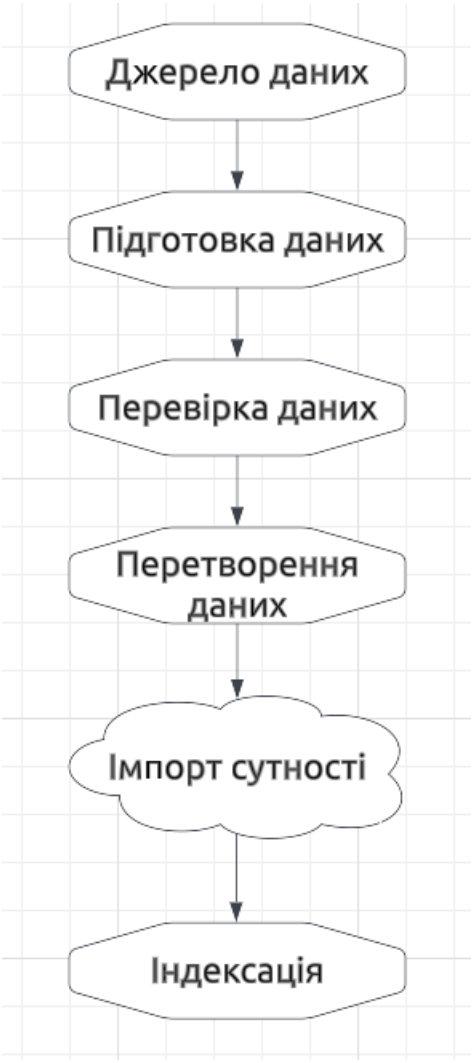


Рисунок 2.4 – Схема процесу трансферу даних для сутностей

Індексація – Magento покладається на індексацію для оптимізації пошуку клієнтів і продуктивності. Після процесу імпорту механізм індексації оновлює індекси пошуку, щоб відобразити щойно імпортовані або оновлені сутності, гарантуючи, що вони доступні для пошуку та доступні для операцій, пов’язаних із сутностями.

Найнеобхідніші процеси імпорту для оптимізації та збільшення продуктивності системи:

- імпорти продуктів,
- імпорти клієнтів,

- імпорти атрибутів та наборів атрибутів,
- імпорти замовлень,
- імпорти запасів та інвентарю.

Інші сутності не здійснюють надмірне навантаження на систему та не потребують вдосконалення трансферу даних.

Кожен з наведених вище процесів імпорту потребує імплементації збільшення продуктивності та оптимізації їх роботи під великі об’єми даних з використанням декмопозиції запитів (рис. 2.4).

Схема процесу трансферу даних для імпорту сутності “продукт” з використанням декомпозиції запиту представлена на рисунку 2.5.

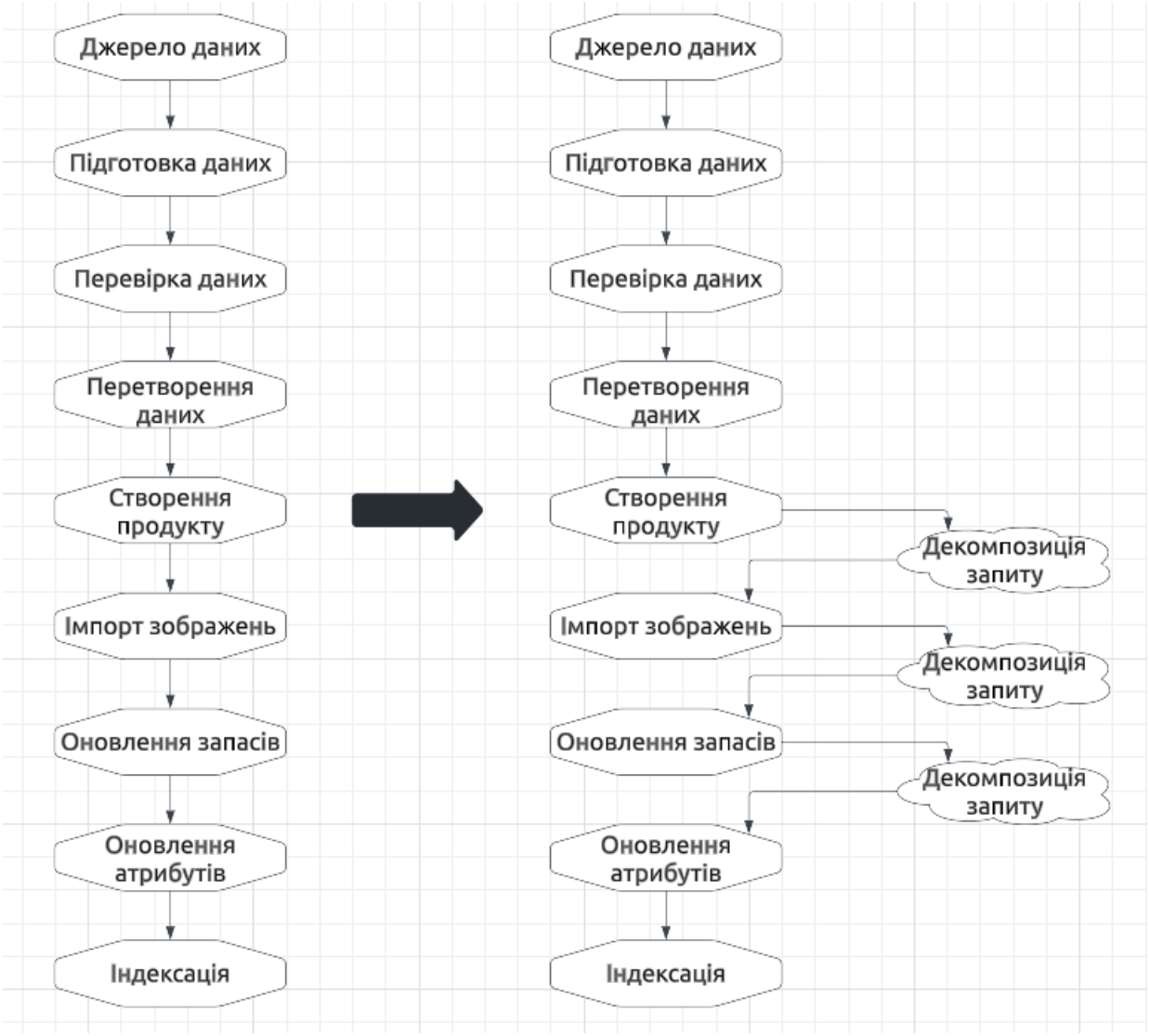


Рисунок 2.5 – Схема процесу трансферу даних для продуктів з використанням декомпозиції запиту

До сутності “продукт” застосовуються наступні операції:

− Створення продукту – перетворені дані використовуються для створення нових продуктів або оновлення існуючих у системі Magento. Цей крок передбачає заповнення атрибутів продукту, налаштування інформації про ціни, призначення категорій і керування варіантами продукту.

− Імпорт зображень – якщо дані продукту містять зображення, на цьому кроці виконується імпорт і зв’язування зображень продукту з відповідними продуктами. Це гарантує, що зображення правильно пов’язані та зберігаються в медіа-галереї Magento.

− Оновлення запасів – якщо імпорт продукту включає кількість запасів, цей крок оновлює рівні запасів для імпортованих продуктів. Він гарантує, що доступність запасів і відповідна інформація синхронізуються в системі інвентаризації Magento.

− Оновлення атрибутів – на цьому етапі виконується оновлення додаткових атрибутів або властивостей продуктів, таких як описи продуктів, атрибути, спеціальні параметри або будь-які інші відповідні дані, пов’язані з продуктами.

Схема процесу трансферу даних для імпорту сутності “клієнт” з використанням декомпозиції запиту представлена на рисунку 2.6.

До сутності “клієнт” застосовуються наступні операції:

− Створення клієнта – перетворені дані використовуються для створення нових облікових записів клієнтів у системі Magento. Цей крок передбачає заповнення атрибутів клієнта, таких як ім’я, електронна адреса, пароль та інша відповідна інформація.

− Імпорт адреси – якщо імпорт клієнта включає інформацію про адресу, на цьому етапі виконується імпорт і пов’язування адрес із відповідними обліковими записами клієнтів. Це гарантує, що деталі адреси правильно пов’язані та збережені в адресній книзі клієнтів Magento.

− Оновлення атрибутів – на цьому етапі виконується оновлення додаткових атрибутів або властивостей облікових записів клієнтів, таких як

спеціальні атрибути, групи клієнтів або будь-які інші відповідні дані, пов’язані з клієнтами.

Рисунок 2.6 – Схема що представляє процес трансферу даних для клієнтів з використанням декомпозиції запиту

Схема процесу трансферу даних для імпорту сутності “атрибути” та сутності “набори атрибутів” з використанням декомпозиції запиту представлена на рисунку 2.7.

До сутності “атрибути” та сутності “набори атрибутів” застосовуються наступні операції:

− Створення атрибутів – перевірені дані використовуються для створення нових атрибутів у системі Magento. Цей крок передбачає визначення властивостей

атрибутів, таких як код, мітка, тип, метод введення та інші відповідні конфігурації.

Рисунок 2.7 – Схема що представляє процес трансферу даних для атрибутів з використанням декомпозиції запиту

− Створення опцій – якщо імпорт атрибутів включає параметри атрибутів (наприклад, значення, що розкриваються, параметри множинного вибору), на цьому кроці обробляється створення та зв’язування параметрів із відповідними атрибутами. Це гарантує, що параметри атрибутів правильно пов’язані та доступні для вибору.

− Оновлення атрибутів – цей крок обробляє оновлення існуючих атрибутів у системі Magento. Це може включати оновлення властивостей атрибутів,

додавання нових параметрів, зміну перевірки введення або будь-які інші необхідні оновлення.

Схема процесу трансферу даних для імпорту сутності “замовлення” з використанням декомпозиції запиту представлена на рисунку 2.8.

Рисунок 2.8 – Схема що представляє процес трансферу даних для змовлень з використанням декомпозиції запиту

До сутності “замовлення” застосовуються наступні операції:

− Створення замовлення – перевірені дані використовуються для створення нових замовлень у системі Magento. Цей крок передбачає заповнення деталей замовлення, таких як інформація про клієнта, адреса доставки, адреса виставлення рахунку та інші відповідні атрибути замовлення.

− Імпорт елементів замовлення – якщо імпорт замовлення включає позиції замовлення (продукти), на цьому кроці виконується імпорт і зв’язування позицій замовлення з відповідними замовленнями. Це гарантує, що продукти правильно пов’язані та включені в замовлення.

− Обробка платежу – на цьому етапі виконується обробка платежу для імпортованих замовлень. Він передбачає обробку платіжних операцій, застосування методів оплати, обчислення підсумків і виконання будь-яких необхідних дій, пов’язаних із платежами.

− Оновлення статусу замовлення – після імпорту замовлення та обробки платежу цей крок оновлює статус замовлення на основі імпортованих даних і результату обробки платежу. Він гарантує, що статус замовлення точно відображає прогрес і виконання замовлення.

Схема процесу трансферу даних для імпорту сутності “запаси” та сутності “інвентар” з використанням декомпозиції запиту представлена на рисунку 2.9.

Рисунок 2.9 – Схема що представляє процес трансферу даних для запасів та інвентарю з використанням декомпозиції запиту

До сутності “запаси” та сутності “інвентар” застосовуються наступні операції:

− Оновлення запасів – підтверджені дані використовуються для оновлення рівнів запасів у системі Magento. Цей крок передбачає оновлення кількості запасів, керування статусом запасів (у наявності або немає в наявності) та обробку будь-яких необхідних коригувань рівнів запасів.

2.2 Проектування структури запитів з використанням принципу декомпозиції

Декомпозиція запиту стосується процесу розбиття складного запиту на менші простіші підзапити, які можна виконувати окремо або паралельно для досягнення бажаного результату.

Результат оптимізатора запитів для декларативного оператора запиту називається планом виконання запиту (QEP). Структура QEP визначає порядок операцій для виконання запиту. QEP зазвичай представлено за допомогою деревоподібної структури, де кожен вузол представляє фізичний оператор бази даних (наприклад, об’єднання вкладеного циклу, сканування таблиці тощо). Для одного запиту може існувати кілька планів, і вибір оптимального плану є головним пріоритетом оптимізатора запитів. Щоб доповнити QEP, більшість оптимізаторів запитів створюють пов’язану з продуктивністю інформацію, таку як інформація про вартість, предикати, оцінки селективності для кожного предиката та статистичні дані для всіх об’єктів, на які посилається оператор запиту.

Зразок плану виконання запиту представлено на рисунку 2.10.

На рисунку 2.10 показано приклад QEP. У цьому QEP дані з чотирьох різних таблиць бази даних (таблиць A, B, C і D) витягуються, фільтруються, об’єднуються, а потім агрегуються для створення бажаних кінцевих результатів. Важливо зауважити, що структура плану, показана на рисунку 2.10, є лише концептуальною структурою, а не фактичним планом від оптимізатора запитів. Він використовується лише для ілюстрації.

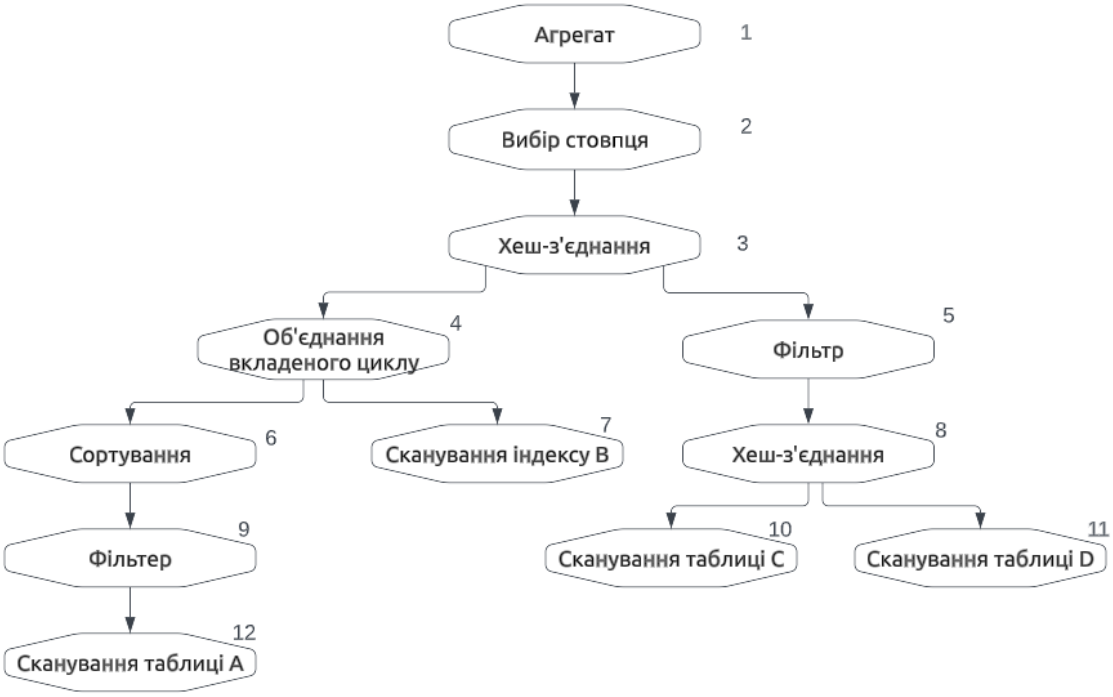


Рисунок 2.10 – Зразок QEP

Оператори в QEP можна класифікувати як оператори блокування або конвеєрні оператори. Оператор блокує, якщо він не видає жодного виходу, доки повністю не використає принаймні один із своїх вводів. Оператори конвеєрної обробки виробляють виходи негайно і безперервно, доки всі вхідні дані не будуть оброблені. Оператор хеш-з’єднання (вузол 8) в

Наприклад, на рисунку 2.10 зображено оператор блокування, а оператори фільтрів (вузли 5 і 9) є операторами конвеєрної обробки, класифікуємо загальні фізичні оператори в QEP наступним чином:

− сканування таблиці, сканування індексу, фільтр, вибір стовпця та об’єднання вкладеного циклу є конвеєрними операторами,

− distinct (або Unique) — блокуючий оператор,

− оператори Sort, Hash Join і Merge-Sort Join блокують,

− Union, Intersect і Except є блокуючими операторами,

− оператори агрегування розглядаються як оператори блокування, хоча насправді вони можуть бути конвеєрними операторами залежно від типу агрегатної функції або від того, чи використовується оператор групування.

Віртуальний вузол — це концептуальний (нефізичний) вузол, який використовується як з'єднувач між двома сегментами у алгоритмі. Він встановлює зв'язок залежності між двома сегментами та реалізований як вузол сканування таблиці, який надає доступ до тимчасової таблиці бази даних. Віртуальний вузол є конвеєрним оператором.

Сегмент — це піддерево QEP таким чином, що (сегменти зображенні на рисунку 2.13):

− кореневий вузол сегмента повинен бути блокуючим вузлом або зворотним вузлом оригінального QEP,

− сегмент може мати щонайбільше один блокуючий вузол,

− усі некореневі вузли в сегменті є конвеєрними вузлами, включаючи віртуальні вузли.

Визначення сегмента гарантує, що будь-який ідентифікований сегмент є максимальною одиницею, яка може бути виконана конвеєрним способом.

Сегменти та віртуальні вузли для QEP представлено на рисунку 2.11.

На рисунку 2.11 показано сегменти та віртуальні вузли для QEP на рисунку 2.10. На цьому рисунку віртуальні вузли I, II та III представляють сегменти I, II та III відповідно. Віртуальний вузол I створює відносини залежності між сегментами I та III. Подібним чином відносини залежності також встановлюються між сегментами II і III віртуальним вузлом II і між сегментами III і IV віртуальним вузлом III.

Сегмент на основі вартості (СВ-сегмент) — це будь-яке дійсне піддерево QEP. На відміну від сегмента, СВ-сегмент не має обмежень. СВ-сегмент доповнюється інформацією про вартість, такою як загальна вартість СВ-сегменту або відсоток вартості СВ-сегмента від загальної вартості QEP. Вартість виражається в одиницях, прийнятих конкретною СУБД. У DB, наприклад,

використовується одиниця під назвою timeron. У роботі CB-сегмент створюється шляхом злиття або декомпозиції сегментів (та/або CB-сегментів).

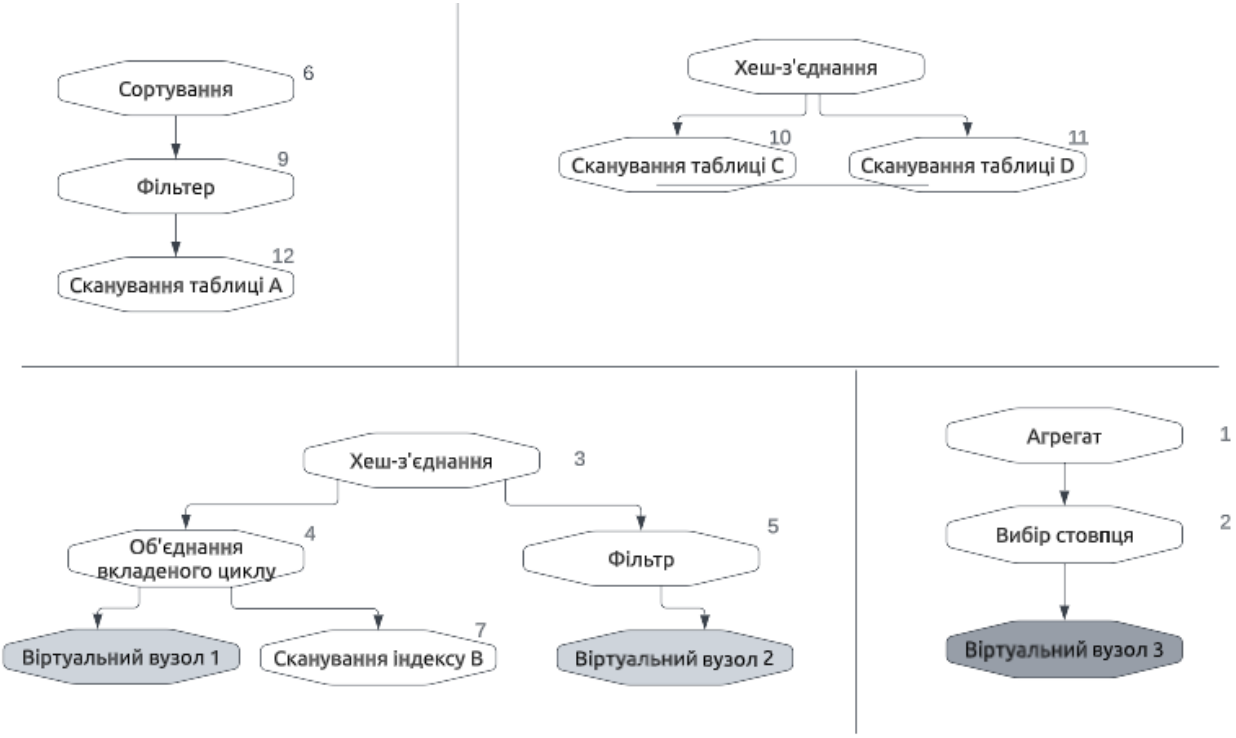


Рисунок 2.11 – Сегменти та віртуальні вузли для QEP на рисунку 2.10

Приклад об'єднання/розкладання сегментів представлено на рисунку 2.12.

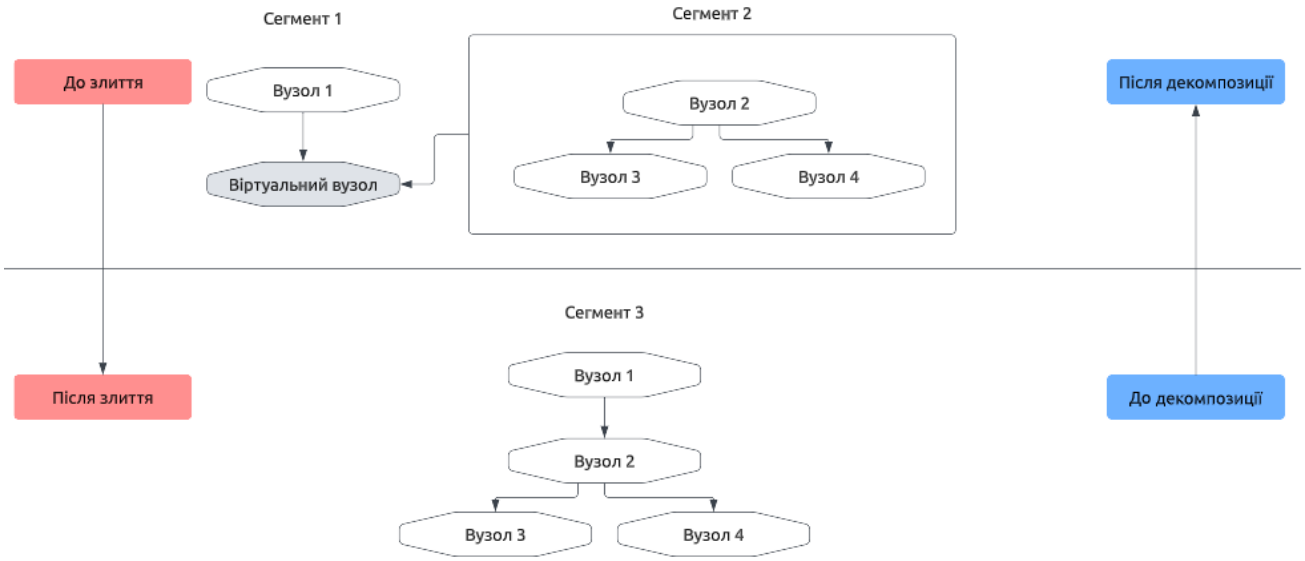


Рисунок 2.12 – Приклад об'єднання/розкладання сегментів

На рисунку 2.12 показано приклад того, як працює процедура злиття/декомпозиції. На цьому рисунку сегмент I і сегмент II можна об’єднати, щоб створити сегмент III, або сегмент III можна розкласти, щоб створити сегмент I і сегмент II. Процес злиття вимагає, щоб сегмент I залежав від сегмента II, тобто сегмент I має віртуальний кінцевий вузол, який представляє сегмент II. Коли сегмент I і сегмент II об’єднуються разом, віртуальний вузол у сегменті I видаляється, а сегмент II додається як дочірнє піддерево батьківського вузла віртуального вузла (у цьому випадку — вузла 1) у початковій позиції віртуального вузла в сегменті I. Новостворене дерево стає об’єднаним сегментом III. Якщо в сегменті, який потрібно об’єднати, існує декілька віртуальних вузлів, тоді кожен віртуальний вузол замінюється сегментом, представленим віртуальним вузлом. Цей процес пояснюється на рисунку 2.12 у напрямку зверху вниз (позначено лінією зліва). Аналогічно, за винятком зворотного напрямку (знизу вгору та позначеного лінією праворуч), коли сегмент III потрібно розкласти, усе піддерево в сегменті III, яке представлене сегментом II, потрібно замінити на віртуальний вузол і таким чином створюється сегмент I. Під час цієї процедури встановлюється зв’язок залежності сегмента між сегментом I і сегментом II.

Залежність сегмента та розклад.

Сегмент може містити віртуальні вузли, а також інші вузли регулярних операцій. Кожен віртуальний вузол у сегменті використовується для представлення іншого сегмента, що означає, що зовнішній сегмент залежить від сегмента, представленого віртуальним вузлом. Якщо сегмент не містить віртуального вузла, він є незалежним. На рисунку 2.11, наприклад, сегменти I і II є незалежними сегментами, оскільки вони не містять віртуальних вузлів. Сегмент III включає два віртуальні вузли, віртуальні вузли I і II, які представляють сегменти I і II відповідно. Отже, сегмент III залежить як від сегментів I, так і від II. Так само сегмент IV залежить від сегмента III, оскільки він містить віртуальний вузол III, який представляє сегмент III. Відносини залежності між сегментами на рисунку 2.11 показані на рисунку 2.13.

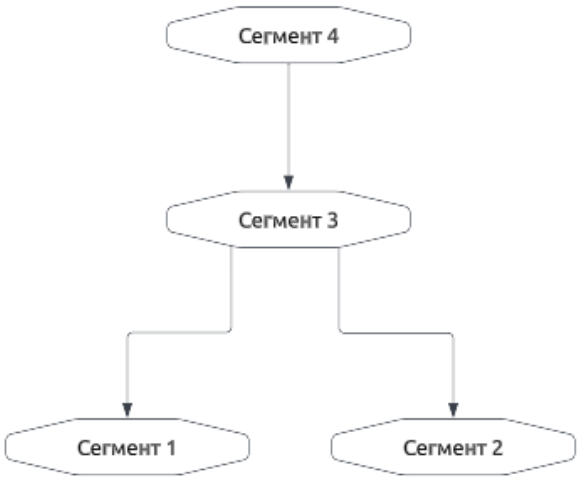


Рисунок 2.13 – Приклад об'єднання/розкладання сегментів

У QEP, якщо сегмент A залежить від сегмента B, тоді підчастина QEP, яка представлена сегментом B, має бути виконана перед підчастиною, представленою сегментом A, оскільки їй потрібен вихід сегмента B для отримання власних результатів. Якщо сегменти незалежні один від одного в QEP, то вони можуть виконуватися в будь-якому порядку або паралельно. У роботі ми визначаємо порядок виконання всіх сегментів у QEP як розклад сегментів для QEP. Базуючись на зв’язках залежності сегментів на рисунку 2.13, графік сегментів для QEP на рисунку 2.10 може бути одним із трьох випадків, показаних на рисунку 2.14 (припускаючи, що паралелізм можливий).

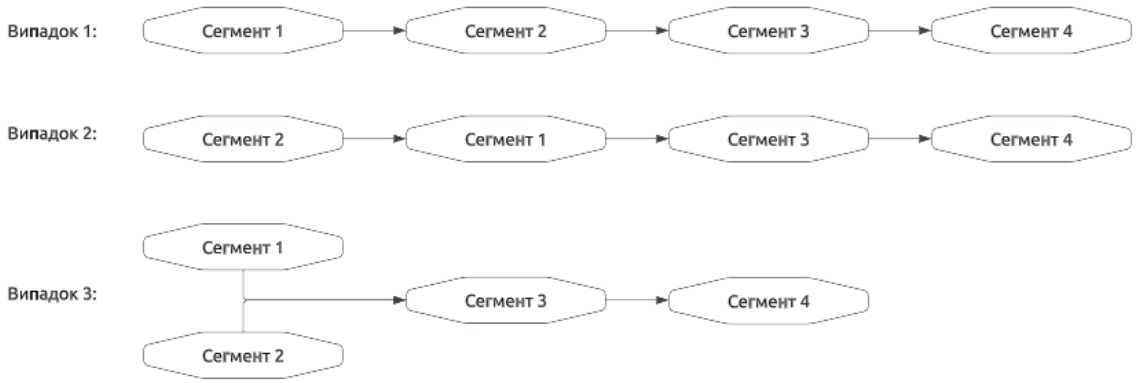


Рисунок 2.14 – Розклад сегментів для QEP на рисунку 2.9

Алгоритм декомпозиції.

Алгоритм декомпозиції спрямований на пошук економічно ефективної стратегії декомпозиції QEP. Алгоритм виконується за два проходи. Перший прохід ідентифікує всі можливі сегменти в QEP шляхом дослідження його деревовидної структури. Під час цього проходу сканування QEP знизу вгору використовується для пошуку вузлів блокування. Кожен блокуючий вузол утворює корінь сегмента, а його нащадки нижчого рівня утворюють піддерево. Як тільки сегмент виявлено, піддерево, яке воно представляє в оригінальному QEP, замінюється віртуальним вузлом, створюючи таким чином нове дерево з віртуальним вузлом як одним із його листів. Ця процедура пошуку та заміни продовжується, доки не буде оброблено всі вузли в оригінальному QEP і не ідентифіковано всі сегменти. У той же час за допомогою віртуальних вузлів, які створюються під час цього проходу, також визначаються відносини залежності між сегментами. Сегмент, що має віртуальний вузол як листовий вузол, залежить від сегмента, представленого віртуальним вузлом.

Перший прохід алгоритму створює набір менших запитів, щоб конвеєрні операції ніколи не переривалися. Однак він не бере до уваги інформацію про вартість. Таким чином, це може генерувати «зміщене» рішення, тобто деякі результуючі сегменти (підзапити) можуть бути дорожчими, ніж інші. «Скошене» рішення має два головних недоліки, через які воно є непрактичним. Перший недолік полягає в тому, що деякі зі згенерованих сегментів самі по собі можуть бути великими, дорогими запитами. Випадок А на рисунку 2.15 показує цю ситуацію. На рисунку 2.15 відсоткове число біля кожного сегмента відображає вартість сегмента як відсоток від загального QEP. Як ми бачимо у випадку А, сегмент III покриває 97% загальних витрат, тоді як інші три сегменти разом покривають решту 3%. У цій ситуації розбиття великого запиту таким чином не вирішить вихідну проблему. Доцільніше розкласти III сегмент далі, якщо це можливо.

Приклад косих рішень представлено на рисунку 2.15.

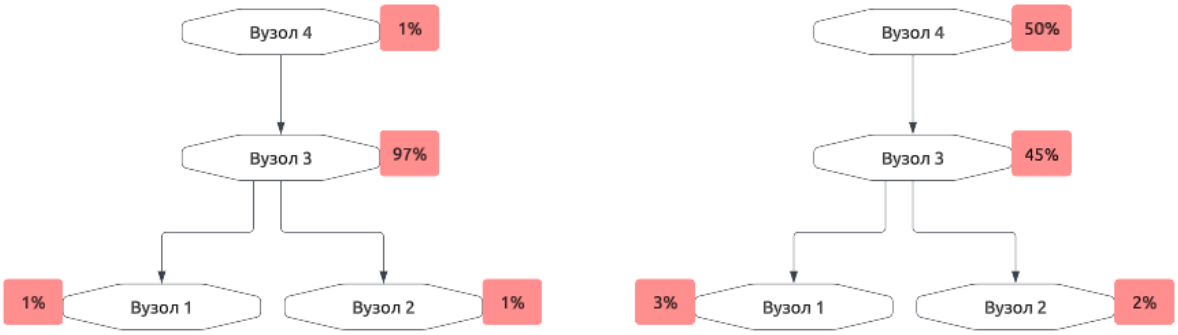


Рисунок 2.15 – Приклади косих рішень

Другий недолік спотвореного рішення полягає в можливості непотрібних накладних витрат на виконання, як це показано у випадку B на рисунку 2.15. Підхід до розкладання великого запиту на кілька менших запитів спричиняє накладні витрати, пов’язані зі зберіганням проміжних результатів. У випадку B на рисунку 2.15 сегменти I та сегменти II покривають дуже невелику частину загальних витрат. Якщо реалізовувати це рішення, проміжні результати з цих двох сегментів зберігаються в тимчасових таблицях, що створює додаткові накладні витрати. У цій ситуації краще об’єднати I і II сегменти в III сегмент.

Щоб подолати недоліки, алгоритм потрібно розширити таким чином, щоб отримати більш збалансоване рішення. На реалізацію цієї мети спрямований другий прохід алгоритму.

Алгоритм завжди намагається досягти збалансованого за вартістю рішення шляхом спочатку об’єднання. Якщо це неможливо зробити, генерується повідомлення, щоб привернути увагу адміністратора бази даних. Отримавши повідомлення, адміністратор може проігнорувати його та просто вважати великий запит таким, що не розкладається, або може вибрати вручну перевірити вузли, а також інформацію про їхню вартість у кожному сегменті, щоб визначити, чи потрібно порушувати чи ні сегментувати далі, щоб досягти більш збалансованого рішення шляхом переривання конвеєрної роботи. Емпіричне правило для цього

ручного процесу полягає в тому, що визначені менші сегменти мають порівну розподіляти вартість вихідного великого сегмента. У алгоритмі адміністративний параметр використовується для керування тим, чи дозволено втручання людини розбивати сегменти вручну.

Варто зауважити, що не завжди можливо розкласти великий запит на збалансоване рішення. Одним із поширених прикладів цієї ситуації є вартість одного вузла (а не сегмента) покриває більшу частину загальної вартості, оскільки алгоритм не обробляє декомпозицію одного оператора, що унеможливлює декомпозицію такого запиту. Такий випадок показано на рисунку 2.16. При застосуванні алгоритму декомпозиції до цього зразка QEP генерується лише один сегмент, оскільки всі оператори в QEP є конвеєрними. Серед усіх цих операторів тільки вузол Table Scan A (вузол 6) покриває майже всю загальну вартість QEP (95%). У цій ситуації навіть із втручанням людини ми не можемо знайти збалансоване рішення. Крім того, застосування декомпозиції в цьому випадку спричиняє непотрібні витрати на виконання. Алгоритм декомпозиції виявляє існування такого випадку та надає відповідний зворотний зв’язок адміністратору баз даних та/або відправнику запиту, щоб вказати, що запит не можна розкласти. Приклад запиту, який не можна розкласти представлено на рисунку 2.16.

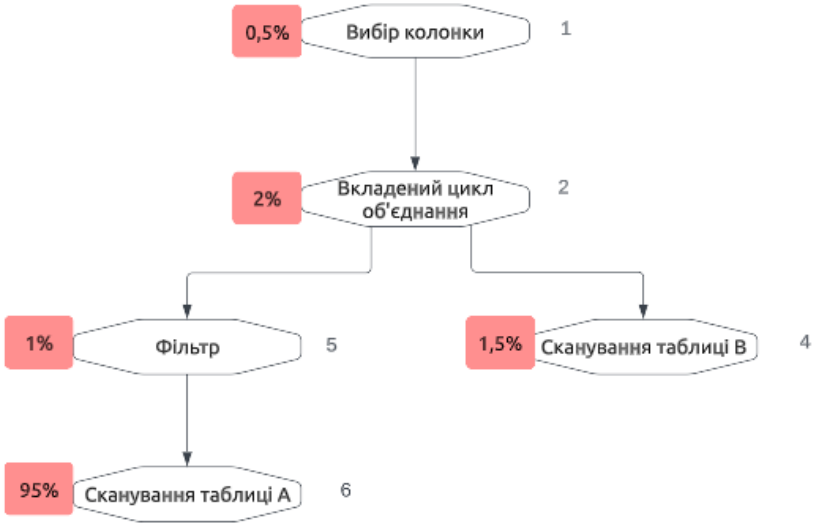


Рисунок 2.16 – Приклад запиту, який не можна розкласти

Фактор викривлення.

Другий прохід процедури декомпозиції використовує «фактор перекосу» (SKF) а рішення, щоб визначити збалансоване рішення.

Припустимо, що для набору відрізків $SEGS = \{seg_1, seg_2 \dots seg_n\}$, відповідний набір витрат становить $COSTS = \{cost_1, cost_2 \dots cost_n\}$, де cost1 — це вартість для seg1, cot2 — вартість значення для seg2 тощо. SKF для SEGS вимірює, наскільки викривлений SEGS з точки зору COSTS. Іншими словами, значення SKF для SEGS вимірює дисперсію COSTS. Чим вище дисперсія, тим вищим має бути значення SKF.

Рисунок 2.17 визначає, як можна обчислити значення SKF для SEGS. У цьому рівнянні VAR представляє дисперсію вибірки, а вартість – це середнє значення COSTS розраховується за рисунок 2.18.

$$SKF\ (SEGS) = VAR\ (COSTS)$$
$$= VAR\ (cost_1, cost_2 \dots cost_n)$$
$$= \sum_{i=1}^{n} (costi - cost')^2 / (n-1)$$

Рисунок 2.17 – Фактор викривлення

$$cost' = MEAN(COSTS)$$
$$= MEAN\ (cost_1, cost_2 \dots cost_n)$$
$$= \sum_{i=1}^{n} costi / n$$

Рисунок 2.18 – Середня COST сегмента

У наведених вище рівняннях сегментні витрати можна вказати як абсолютні значення або як відносні значення. Відносна вартість сегмента в SEGS визначається як відсоток абсолютної вартості сегмента до загальної абсолютної вартості всіх сегментів у SEGS. Перевага використання значень відносної вартості

в рівняннях 1 і 2 (відповідно рисунок 2.17 та рисунок 2.18) полягає в тому, що воно нормалізує розраховане значення SKF до діапазону [0, 1]. Без нормалізації неможливо легко вказати загальне порогове значення SKF, яке може використовуватися другим проходом алгоритму декомпозиції для пошуку збалансованого за вартістю рішення. Значення SKF, може коливатися в широких межах залежно від запиту та способу декомпозиції запиту.

У підході відносні витрати на сегменти використовуються для розрахунку вартості SKF. Значення адміністративного порогу за замовчуванням установлено як 0,07, що відповідає розподілу витрат «30% проти 70%» у 2-сегментному рішенні, що означає, що великий запит можна розкласти на 2 менших сегменти, один із яких охоплює 30% загальну вартість, а інший покриває 70% загальної вартості. Будь-яке рішення, значення SKF якого перевищує порогове значення, розглядається алгоритмом як спотворене рішення, і тому його необхідно об’єднати (або розкласти) далі.

2.3 Проектування інформаційної системи

Діаграма послідовностей для процесу імпорту даних представлена на рисунку 2.19.

Актором у діаграмі є адміністранор (користувач системи), який запускає процес іморту нажавши на кнопку старт. Далі, за допомогою технології Ajax, до контролера POST-методом відправляються параметри типу імпорту. Далі ідентифікатор типу, який входить до параметрів, передається до моделі, щоб розпочати імпорт сутності. У моделі для ідентифікатора відправляється запит до хелпера на отримання даних до БД Magento. Хелпер складає запит до БД. Після отримання даних від хелпера, модель додає інформацію до логів про те, що розпочинається імпорт сутності та розпочинається етап імпорту даних. Результатом етапу є оновлення/створення/видалення в БД записів з джерела імпорту. Інформація про зміни в даних логується та далі запускається метод перевірки цих даних. Перевірка складається з зіставлення кількості змінених

записів. Далі результати перевірки логуються, модель повертає результат імпорту сутностей до контролера. Контролер формує JSON з отриманої інформації та надсилає до представлення, де інформація дешфрується та форматується, і відображається користувачу у кінцевій формі.

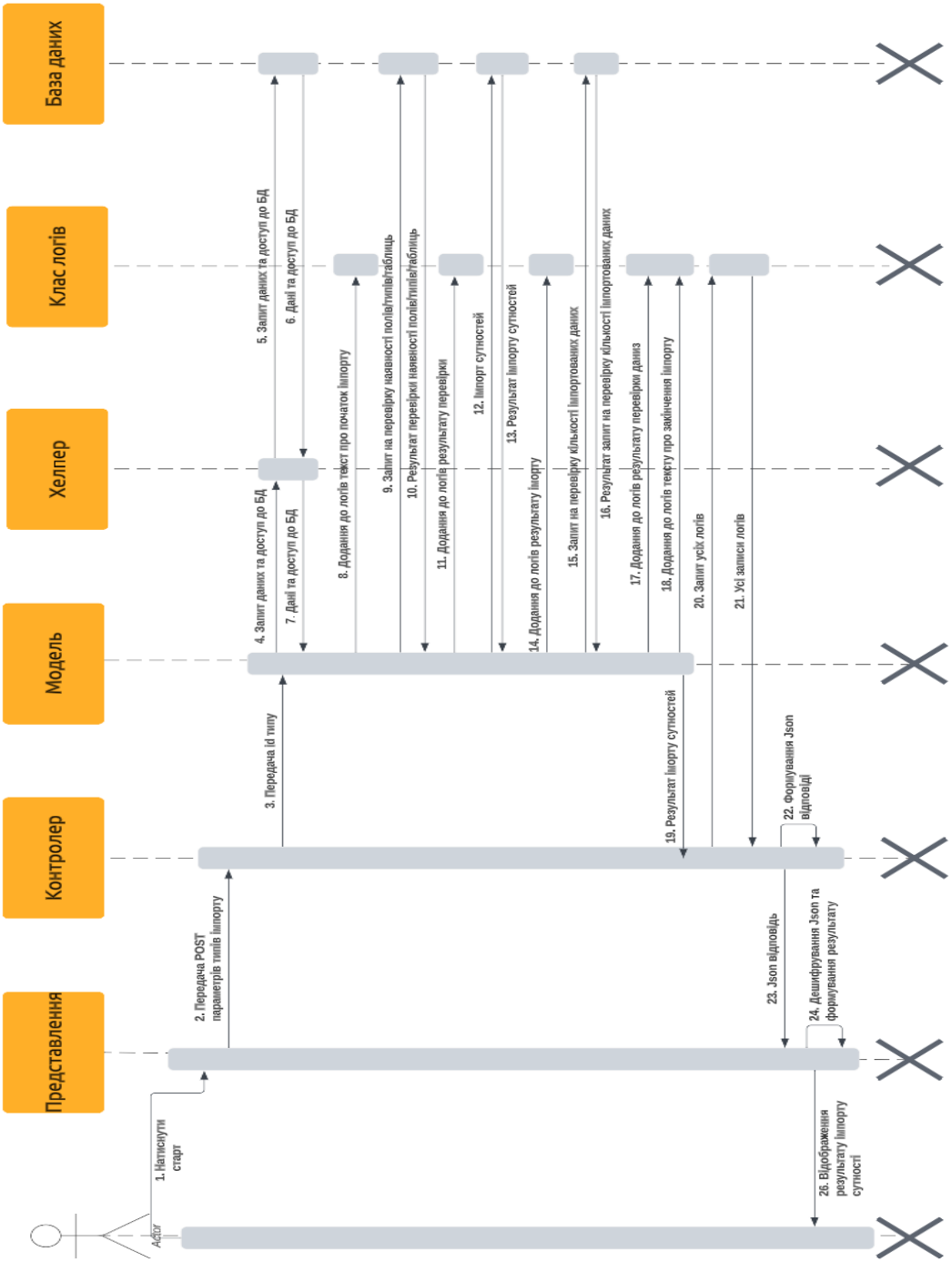


Рисунок 2.19 – Діаграма послідовностей для процесу імпорту даних

На першому рівні деталізації описана основна функція системи – імпорт сутності. На вхід до даної функції відправляється інформація про сутність та тип імпорту. В якості керуючого параметру виступають внутрішні правила компанії та внутрішні правила системи. Механізмом керування є програмна система та адміністратор системи, котрий запускає процес імпорту. В результаті отримуємо імпортовані дані та логи про результат імпорту. Діаграма модуля імпорту сутностей представлена на рисунку 2.20.

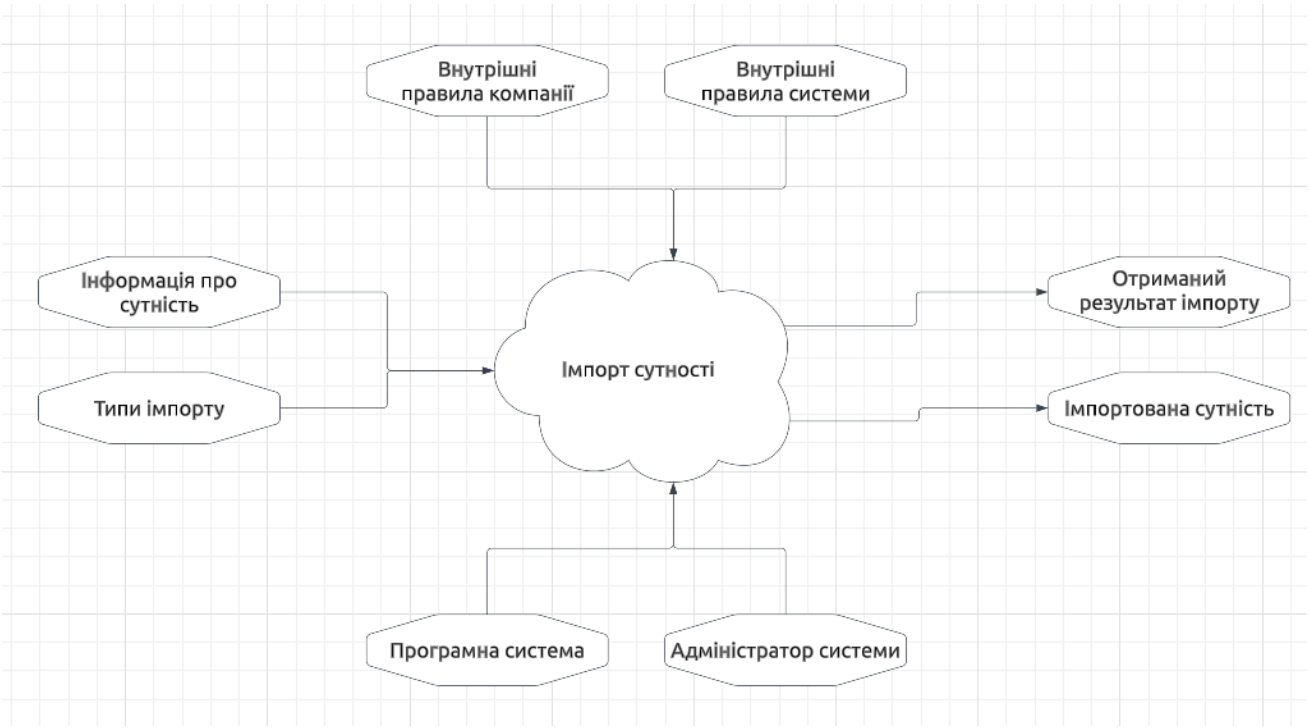


Рисунок 2.20 – Діаграма модуля імпорту сутностей

При розбитті складного процесу на складові його функції використовується принцип декомпозиції.

На рисунку 2.21 представлена діаграма декомпозиції міграції бази даних, на якій зображене розбиття головної функції модуля на чотири менші за об’ємом функції, а саме:

- ініціалізація параметрів,
- запуск етапу міграції,
- формування відповіді користувачу,
- відображення результата користувачу.

Інформація про сутність необхідна для ініціалізації параметрів. На цю функцію впливають адміністратор системи, який заповнює форму налаштування необхідними даними та розпочинає процес імпорту, та програмна система. Управління дають правила компанії та системи. На наступні функції вже впливає програмна система, а внутрішні правила цієї системи надають управління.

Для початку процесу типу іморту необхідні параметри системи, які можуть бути отримані на етапі ініціалізації. Формування відповіді користувачу форматується у JSON на основі отриманих даних після завершення процесу імпорту сутності. Далі відповідь проходить етап обробки та відображається кінцевому користувачеві.

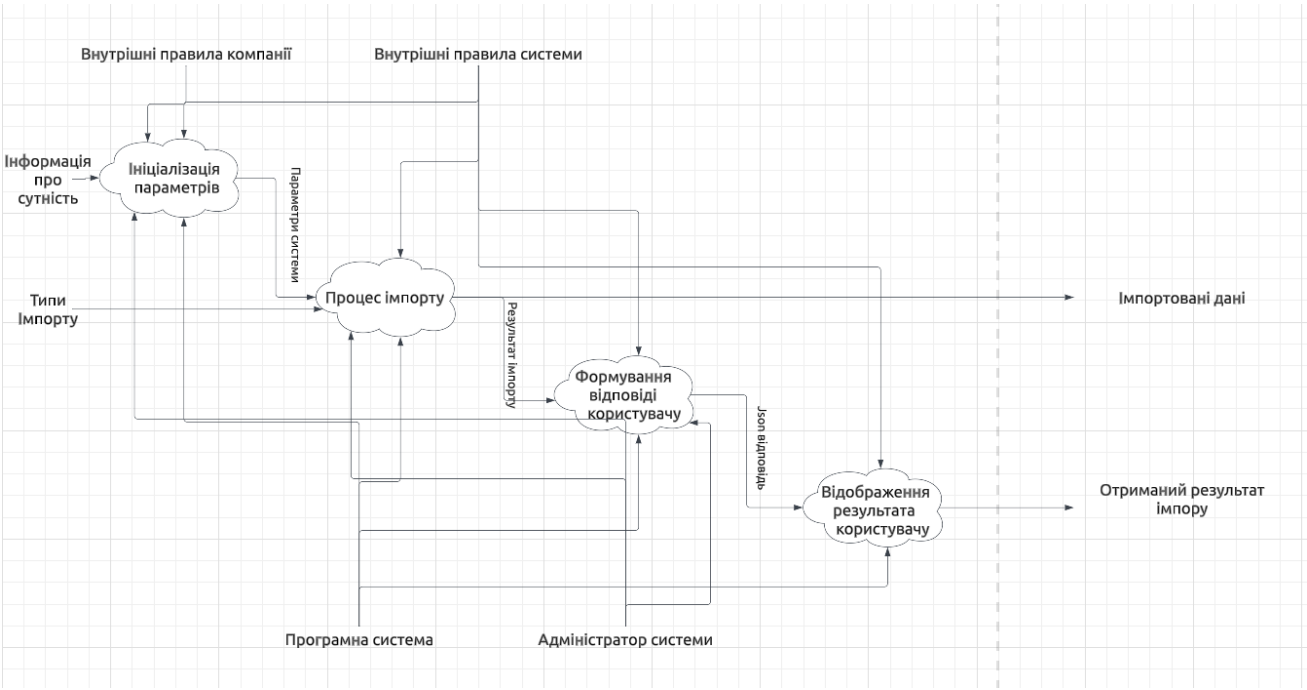


Рисунок 2.21 – Діаграма декомпозиції імпорту сутностей

Процес імпорту декомпозується на:

- вибір типу імпорту,
- процес імпорту даних;
- перевірка імпортованих даних.

Діаграма декомпозиції процесу імпорту сутності показана на рисунку 2.22

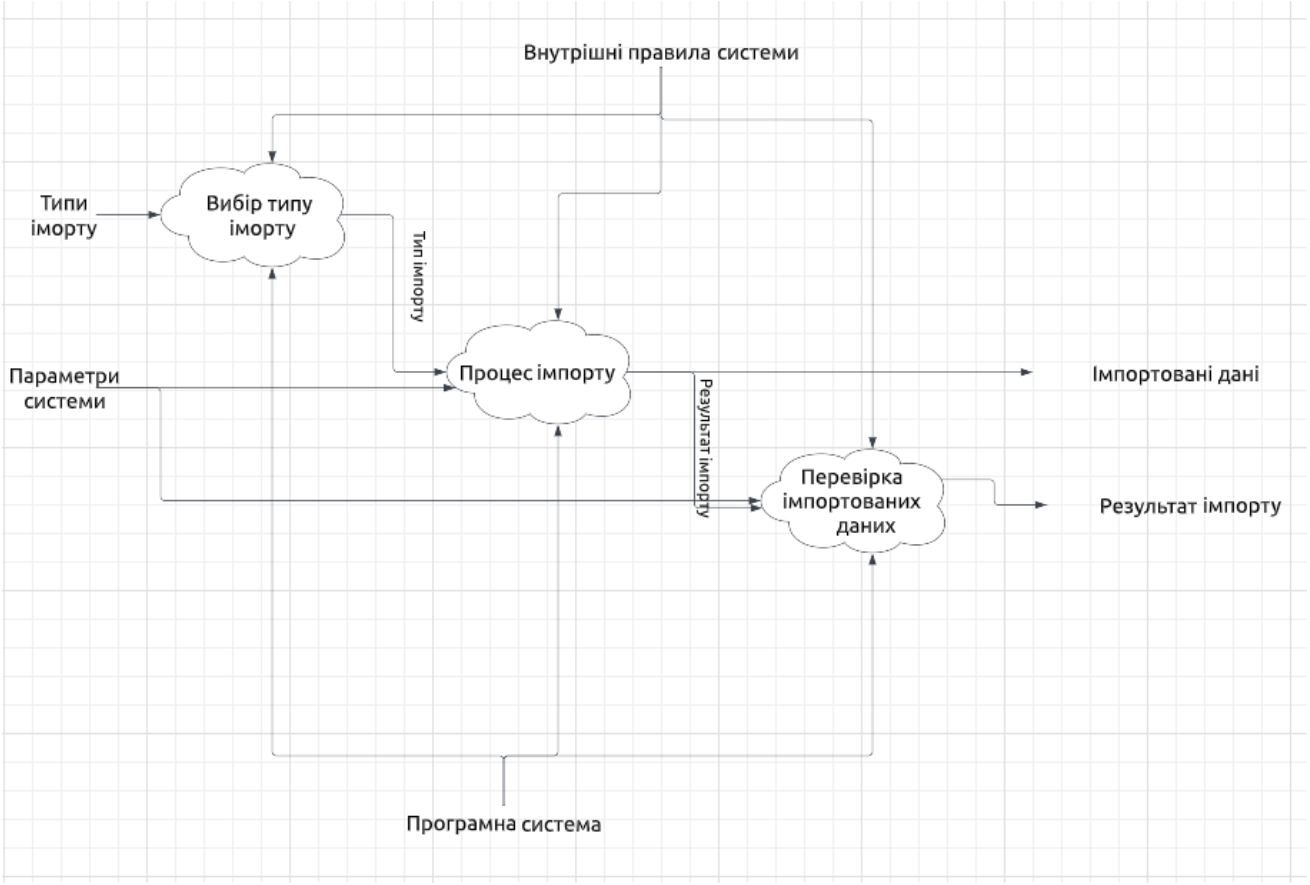


Рисунок 2.22 – Діаграма декомпозиції процесу імпорту сутності

На всі підпроцеси процесу імпорту впливає програмна система, а для управління використовуються внутрішні правилами системи. Вхідними параметрами функції вибору типу імпорту є безпосередньо дані про типи іморту. На виході отримуються дані про поточний тип імпорту. Якщо імпорт даних можливий, розпочинається процес імпорту. Результат імпортованих даних є вхідним для функції перевірки імпортованих даних, та параметри системи. На основі цих даних формується результат імпорту, який використовується для формування відповіді користувачу.

Процес формування відповіді користувачу розкладається на наступні етапи:

- збір логів успішних етапів,
- збір логів попереджень,
- збір логів помилок,
- обробка даних та шифрування в JSON-формат.

Діаграма декомпозиції формування відповіді користувачу представлена на рисунку 2.23

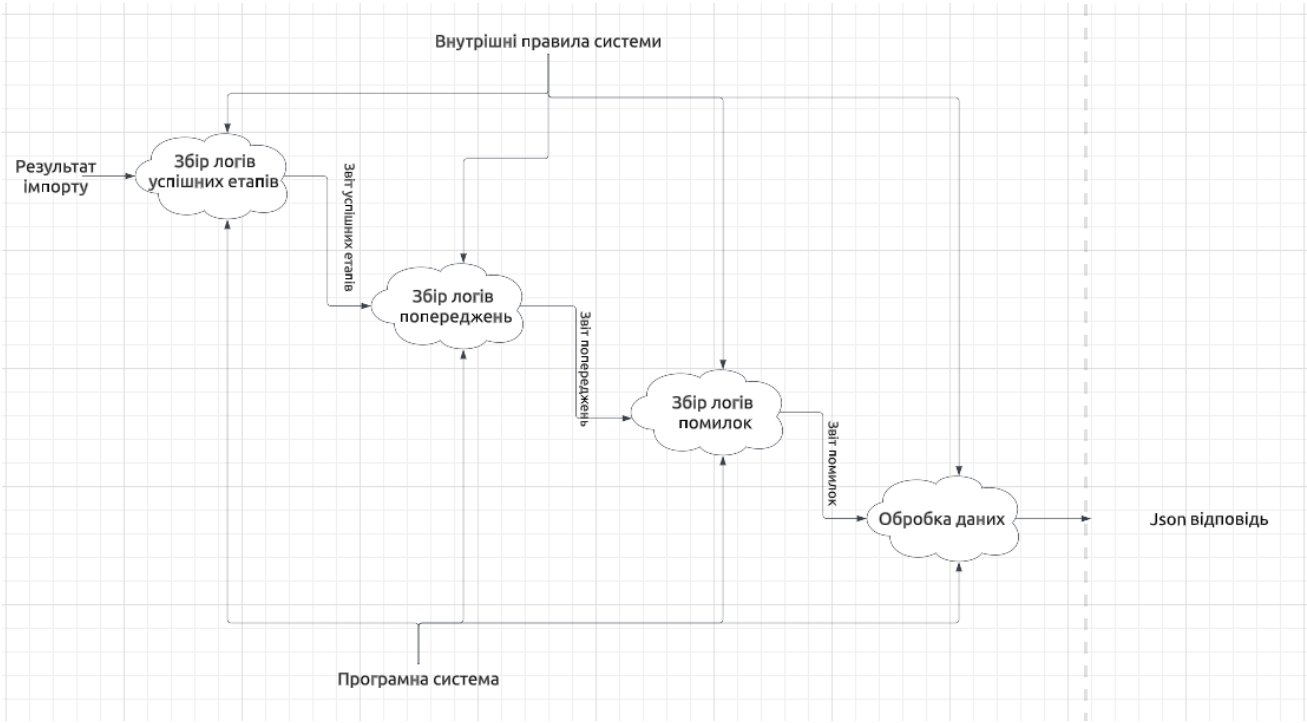


Рисунок 2.23 – Діаграма декомпозиції формування відповіді користувачу

На всі підпроцеси процесу формування відповіді впливає програмна система, управління виконують внутрішніми правилами системи. В процесах збору логів використовується інформація про результати імпорту, обробляється та формується у масив звітів, який форматується у JSON-формат для відображення користувачеві.

Процес відображення результата поділяється на три функції:

- дешифрування JSON-даних,
- заповнення текстового поля даними логів,
- змінення строки стану.

На рисунку 2.24 описана діаграма декомпозиції відображення результата користувачу

Отримана JSON-відповідь дешифрується та направляється для заповнення текстового поля для логів, далі виділяються дані про стан процесу імпорту та

змінюється рядок стану. Наприкінці користувач може отримати результат виконання імпорту сутності.

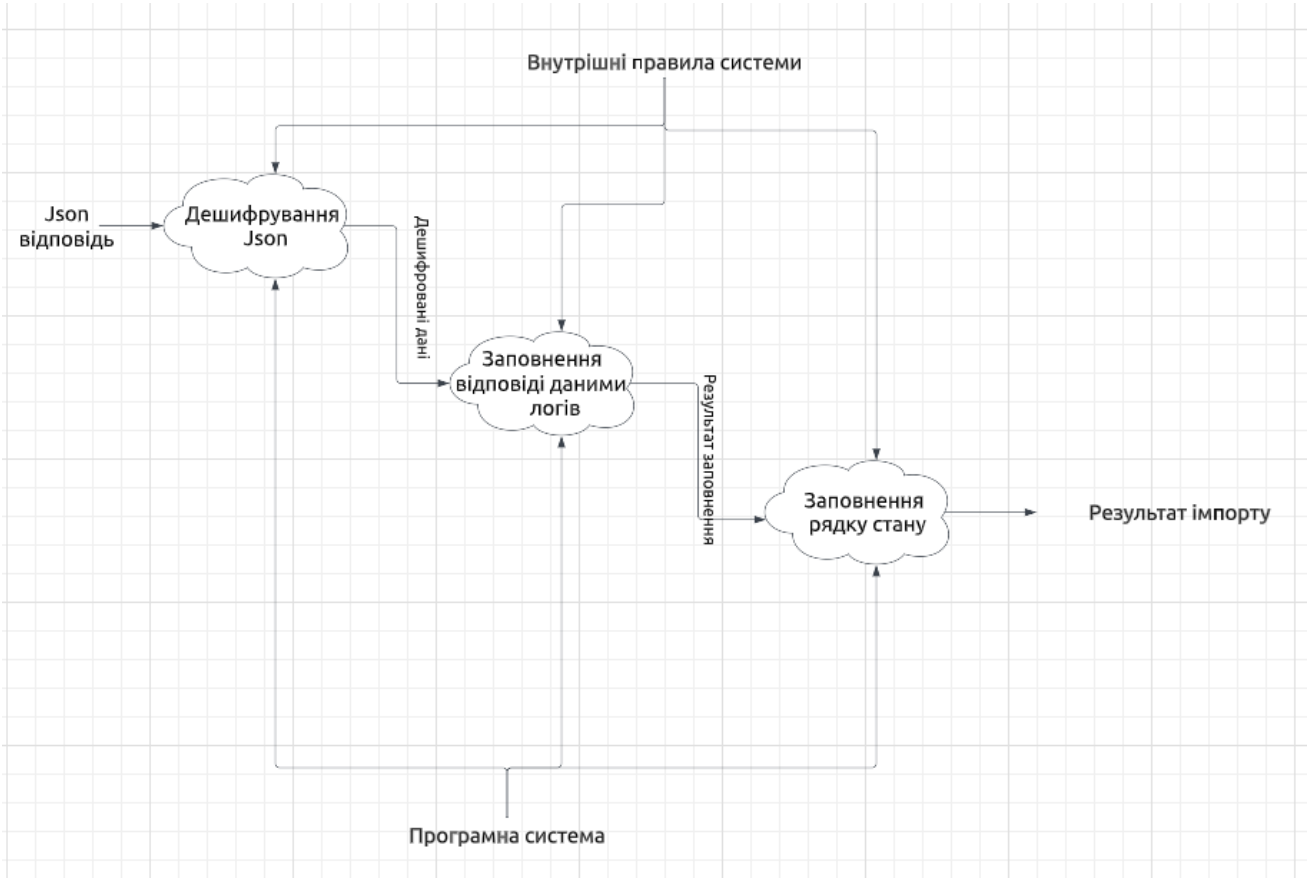


Рисунок 2.24 – Діаграма декомпозиції відображення результата користувачу

2.4 Висновки до розділу 2

Реалізоваго технологію передачі даних для платформи електронної комерції Magento 2 за допомогою декомпозиції запитів. Декомпозиція запитів передбачає розбиття складних запитів на менші підзапити для більш ефективного виконання та обробки. Розбиття складного запиту на менші, більш керовані підзапити, забезпечує паралельне виконання, розподілену обробку та покращену масштабованість. Розкладені запити можна розподілити між кількома ресурсами, що забезпечує горизонтальну масштабованість, оскільки для обробки підзапитів можна додати більше ресурсів. Декомпозиція запитів полегшує оптимізацію запитів на детальному рівні, дозволяючи ефективно використовувати методи індексування, кешування та перезапису запитів.

Під час декомпозиції запитів забезпечується узгодженість даних у різних підзапитах та їхніх результатах, механізми обробки помилок реалізовані для обробки помилок, які виникають під час виконання підзапитів, і для забезпечення цілісності загальних результатів запиту.

Розділ 3 Програмна реалізація технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів

3.1 Особливості декомпозиції запитів при реалізації технології трансферу даних для платформи Magento 2 e-commerce

Опис псевдокоду першого проходу алгоритму декомпозиції показано на рисунку 3.1. Він включає ітераційну процедуру під назвою «FindSegments», яка застосовується до QEP запиту. Після виконання ітераційного процесу цієї процедури всі сегменти в QEP, а також їхні зв'язки залежностей ідентифікуються та зберігаються у двох глобальних наборах, GSegSet і GSegRelSet.

Перший прохід алгоритму декомпозиції представлено на рисунку 3.1.

```
1   FindSegments (QEP) {
2       subTreeSet ← create a empty tree set;
3       FOR each leaf node (curLeaf) of QEP
4           curPath ← identifies the tree path that curLeaf belongs to in QEP;
5           cbaNode ← find its closet blocking ancestor node or the return node along curPath;
6           subTree ← create a sub-tree that is rooted at cbaNode and includes all nodes in the
                curPath from cbaNode to curLeaf;
7           add subTree into subTreeSet;
8       ENDFOR;
9       REPEAT
10          curSubTree ← get next sub-tree from subTreeSet;
11          newSeg ← NULL;
12          IF the root node of curSubTree has only one input in the original QEP THEN
13              newSeg ← create a new segment that has the same structure as curSubTree;
14          ELSE
15              matchedSubTrees ←find other sub-trees that have the same root as curSubTree;
16              IF matchedSubTrees is not empty THEN
17                  newSeg ← create a new segment by merging matchedSubTrees with curSubTree
                        such that shared nodes only appear once in the segment;
18                  remove matchedSubTrees from subTreeSet;
19              ENDIF;
20          ENDIF;
21          IF newSeg is not NULL THEN
22              add newSeg into GSegSet;
23              update QEP such that the whole newSeg sub-tree in QEP is replaced by a newly
                    created virtual node;
24          ENDIF:
25          remove curSubTree from subTreeSet
26      UNTIL subTreeSet is empty;
27      REPEAT
28          call procedure "FindSegments(newQEP)";
29          segRels ← create segment dependency relationships between any two segments that are
                found in two consecutive iterations if they are connected by a virtual node;
30          add segRels into GSegRelSet;
31      UNTIL all nodes in the original QEP are processed
32  }
```

Рисунок 3.1 – Алгоритм декомпозиції – перший прохід

На рисунку 3.2 показано псевдокод для цього проходу.

```
1   ReOrganizeSegments (QEP) {
2       call procedure "FindSegments(QEP)" to generate the global segment set "GSegSet" and the
            global segment relationship set "GSegRelSet";

3       FOR each segment(curSeg) in GSegSet
4           calculate the cost for curSeg based on QEP compiler information;
5       ENDFOR;

6       curSKF ← calculate the "skew factor" for GSegSet;
7       validSKFRange ← a pre-defined acceptable SKF range;

8       REPEAT
9           minSeg ← find the smallest segment (having minimum cost) in GSegSet.;
10          conSeg ← find the smallest segment that is connected to minSeg in GSegRelSet (either
                depends on or is depended on minSeg);
11          newCBSeg ← merge minSeg and conSeg to create a new larger CB-segment;
12          update GSegSet such that minSeg and conSeg are removed and newCBSeg is added;
13          update GSegRelSet such that all segment dependency relationships that involve minSeg
                and conSeg are modified correctly to involve newCBSeg instead;
14          curSKF ← calculate the "skew factor" for GSegSet;
15      UNTIL curSKF is within validSKFRange OR there is only one segment left in GSegSet;

16      IF there is only one segment left in GSegSet THEN
17          notify "Exception Management" module that a cost balanced solution is impossible
18      ENDIF;
19  }
```

Рисунок 3.2 – Алгоритм декомпозиції – другий прохід

Псевдокод на рисунок 3.2 визначає, як досягається збалансоване за вартістю рішення шляхом об'єднання сегментів, знайдених під час першого проходу алгоритму. Він не враховує сегменти, що розкладаються. Використовуючи алгоритм, розкладання великого сегмента завжди перериває конвеєрну операцію.

3.2 Результати декомпозиції запитів при реалізації технології трансферу даних для платформи Magento 2 e-commerce

Порівняння результатів продуктивності між звичайним SQL-запитом і SQL-запитом з декомпозицією не є простим і залежить від різних факторів, таких як складність запиту, передача даних, індексування, оптимізація, вимоги до узгодженості даних і масштабованість.

Під час проведення тестувань продуктивності з використанням декомпозиції зібрано наступні результати відображенні на рисунку 3.3 та на рисунку 3.4, де вертикальна шкала – пропускна здатність (запит/секунду), а горизонтальна шкала інтервали часу.

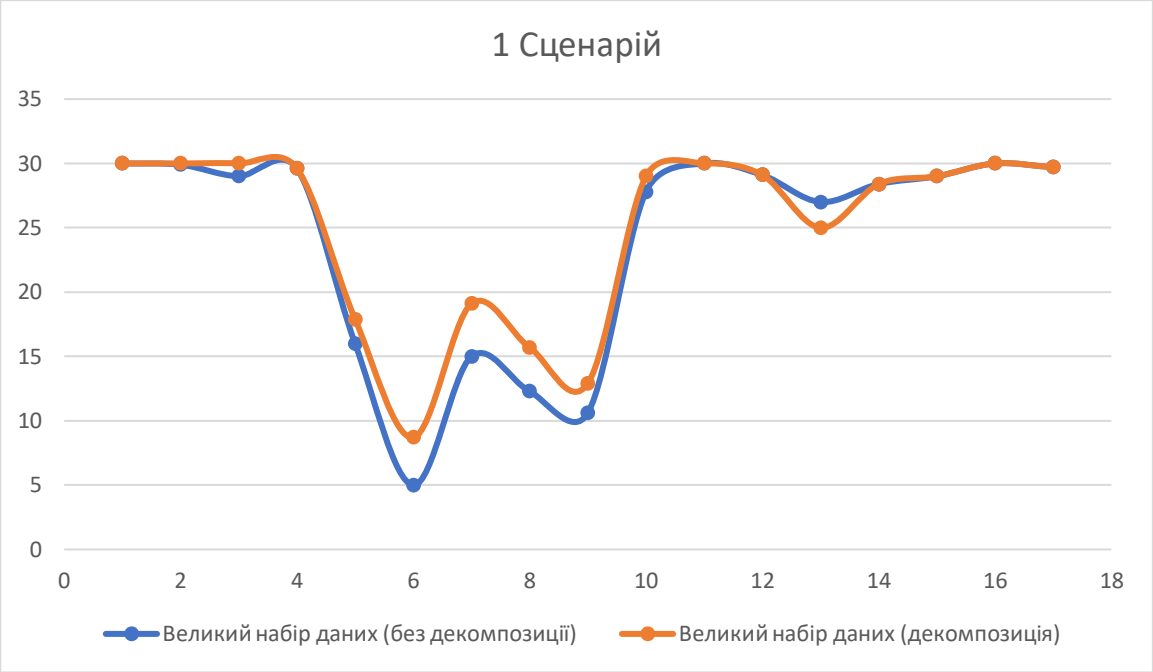


Рисунок 3.3 – Графік порівняння продуктивності запитів, сценарій 1

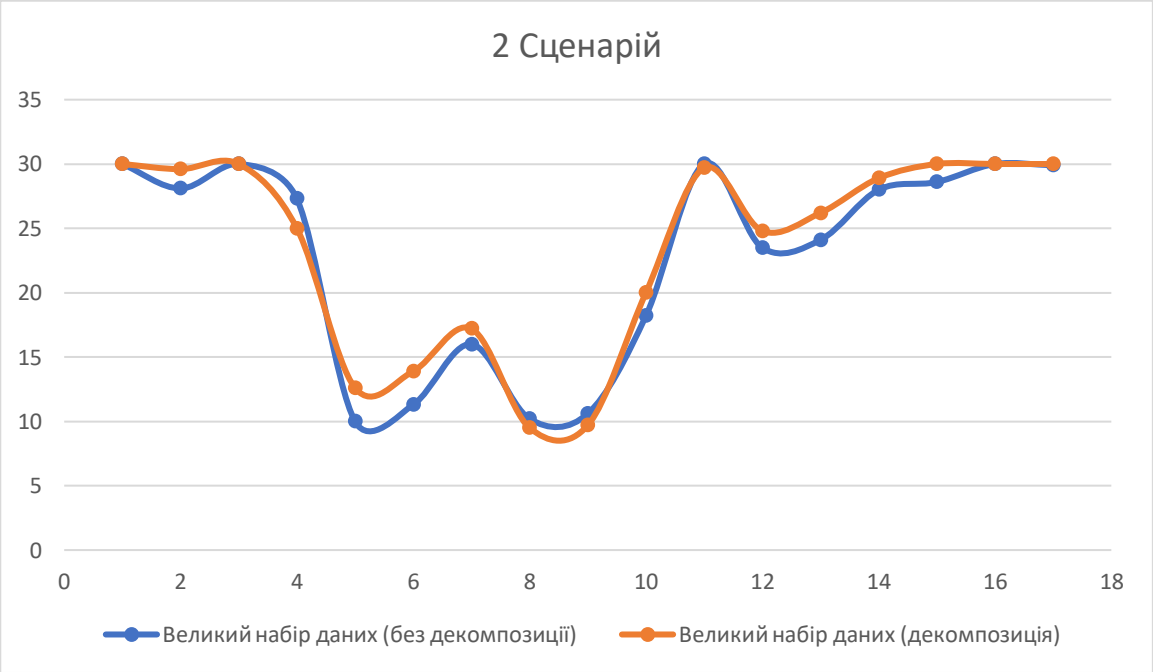


Рисунок 3.4 – Графік порівняння продуктивності запитів, сценарій 2

Підрезюмувавши графіки, отримані результату дають 9.2% у сценарії 1 та 6.6% у другому сценарії (результати застосування декомпозиції запитів відображено на таблиці 3.1), збільшення продуктивності під час виконання великих запитів. Незалежно від 1 чи 2 сценарію, у більшості випадків результатом декомпозиції запитів є збільшення пропускної здатності, тим самим пришвидшення виконання маніпуляцій з базою даних.

Таблиця 3.1 – Результати застосування декомпозиції запитів

	Середня пропускна здатність (без декомпозиції)	Середня пропускна здатність (з декомпозицією)	Збільшення продуктивності (%)
Сценарій 1	24.024	24.947	9.2%
Сценарій 2	22.694	23.359	6,6%

3.3 Опис функціональних можливостей інформаційної системи

Створене розширення – це багатофункціональний модуль розширення Magento 2. Він забезпечує комплексні функції імпорту та експорту для ефективного керування даними на платформі електронної комерції Magento 2. Цей модуль пропонує широкий спектр розширених функцій, зручний інтерфейс і надійну оптимізацію продуктивності.

Ключові особливості:

− кілька форматів файлів – модуль підтримує різні формати файлів для імпорту та експорту даних, включаючи CSV, XML, JSON, ODS, XLSX тощо (рис. 3.1). Ця гнучкість дозволяє працювати з різними джерелами даних і легко інтегруватися із зовнішніми системами (представлено на рисунку 3.5),

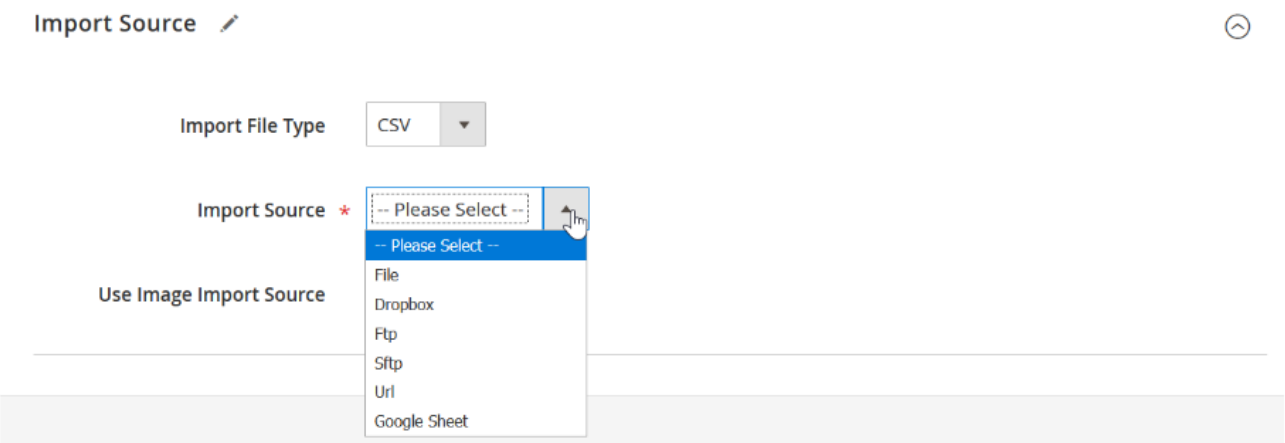


Рисунок 3.5 – Формати файлів для імпорту та експорту

– розширена конфігураці – модуль дозволяє зіставляти поля ваших файлів імпорту/експорту з відповідними атрибутами Magento 2. Ця функція спрощує процес передачі даних і забезпечує точне відображення даних між різними системами,

– пакетна обробка – модуль підтримує пакетну обробку, дозволяючи імпортувати та експортувати великі набори даних у менших, керованих фрагментах. Ця функція оптимізує використання пам’яті та покращує загальну продуктивність,

– історія імпорту та журнали – модуль зберігає детальну історію всіх операцій імпорту, надаючи таку інформацію, як статус імпорту, час завершення та будь-які виявлені помилки. Допомагає функція комплексного журналу з усуненням несправностей і відстеження імпорту (представлено на рисунку 3.6),

– перевірка даних – модуль пропонує вбудовану функцію перевірки даних, щоб забезпечити цілісність і точність даних під час процесу імпорту. Це дозволяє налаштувати правила перевірки та параметри обробки помилок, щоб забезпечити високу якість даних (представлено на рисунку 3.7),

– обробка атрибутів – модуль надає додаткові параметри для керування атрибутами під час операцій імпорту та експорту. Ви можете створювати нові атрибути, оновлювати існуючі, обробляти параметри атрибутів і ефективно керувати значеннями атрибутів (представлено на рисунку 3.8),

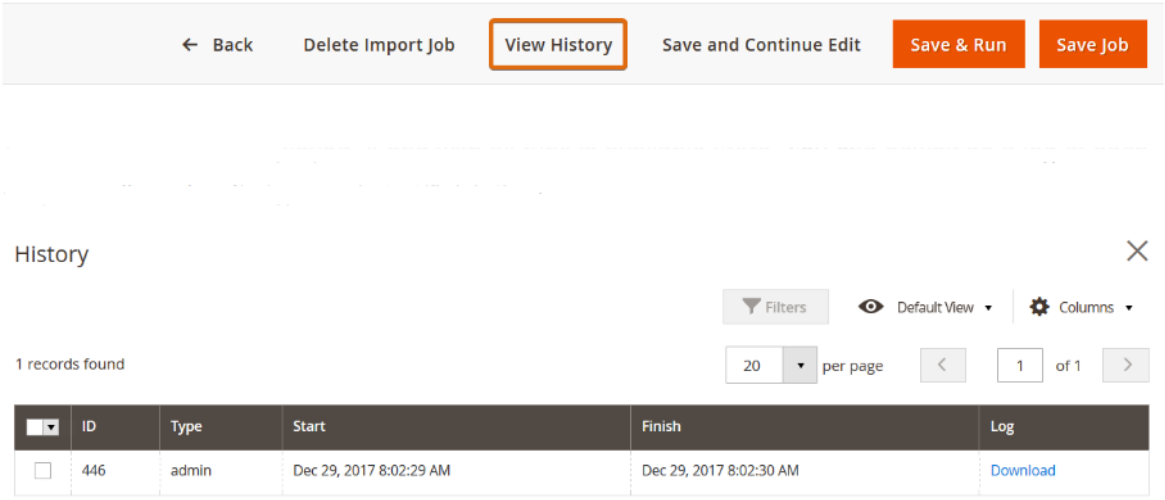


Рисунок 3.6 – Історія імпорту та журнали

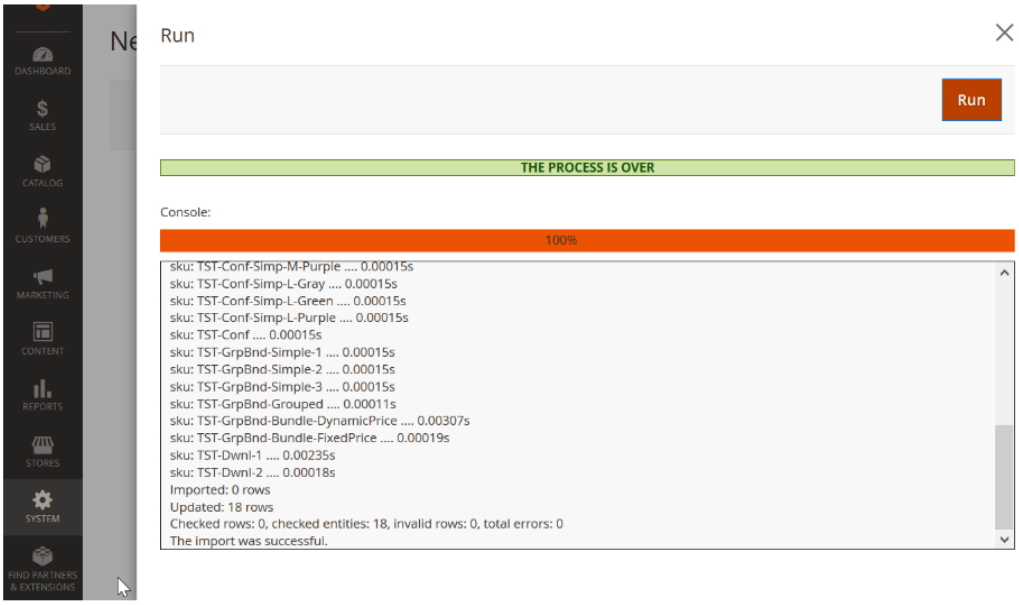


Рисунок 3.7 – Перевірка даних

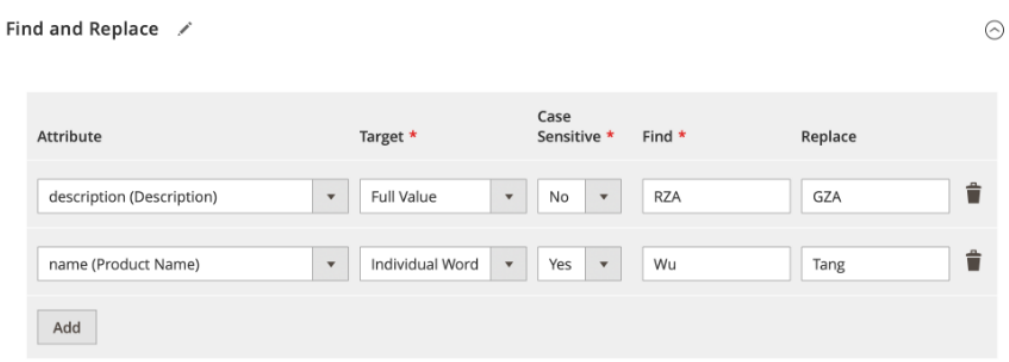


Рисунок 3.8 – Обробка атрибутів

– імпорт категорій – модуль підтримує операції імпорту та оновлення категорій. Ви можете легко імпортувати та оновлювати інформацію про категорії, включаючи ієрархічні структури та спеціальні атрибути, що спрощує завдання керування категоріями (представлено на рисунку 3.9),

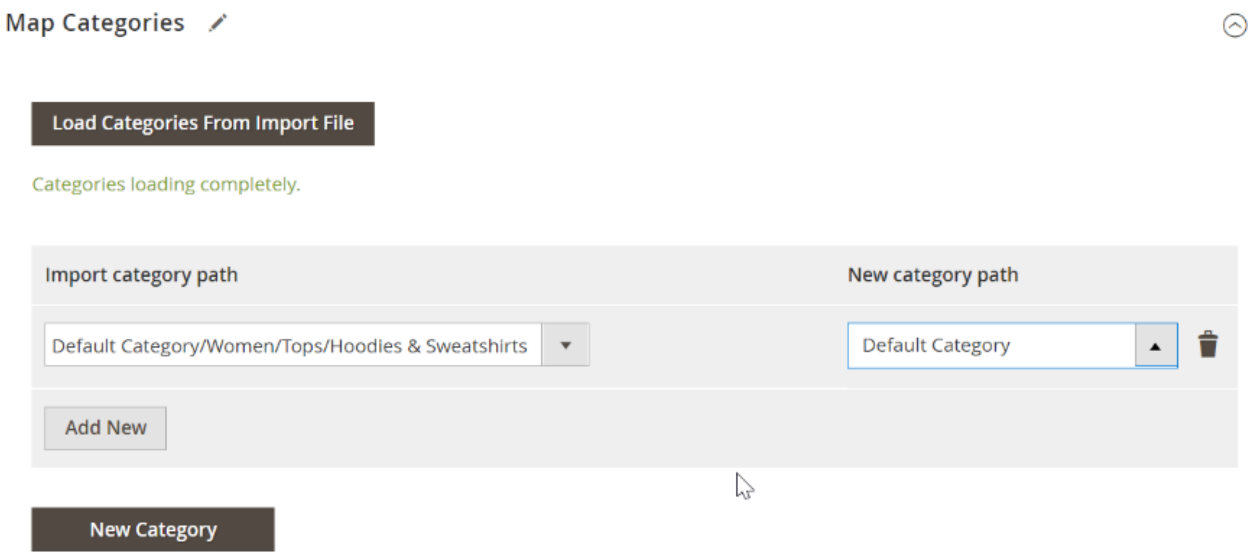


Рисунок 3.9 – Імпорт категорій

– профілі імпорту – модуль дозволяє зберігати конфігурації імпорту як профілі для подальшого використання. Ця функція дає змогу повторно використовувати налаштування імпорту та спрощувати повторювані завдання імпорту, заощаджуючи час і зусилля,

– інтеграція API – модуль пропонує інтеграцію з API Magento 2, що дозволяє виконувати операції програмного імпорту та експорту. Ця інтеграція забезпечує гнучкість і можливості автоматизації для синхронізації даних (представлено на рисунку 3.10),

– запланований імпорт – можна запланувати автоматичний запуск завдань імпорту через заздалегідь визначені проміжки часу. Ця функція корисна для регулярних завдань синхронізації даних, забезпечуючи узгодженість і точність даних у всіх системах (представлено на рисунку 3.11),

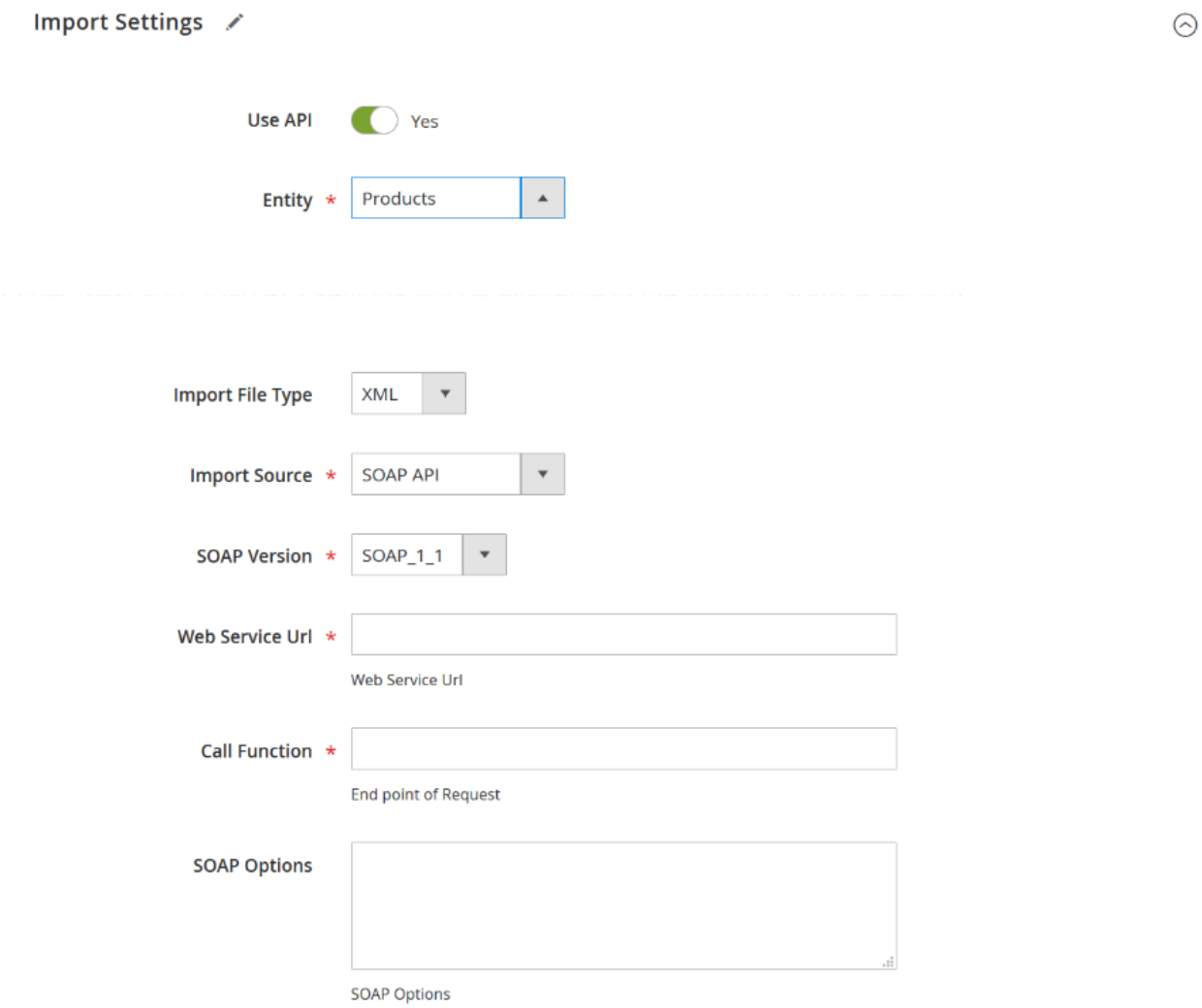


Рисунок 3.10 – Інтеграція API

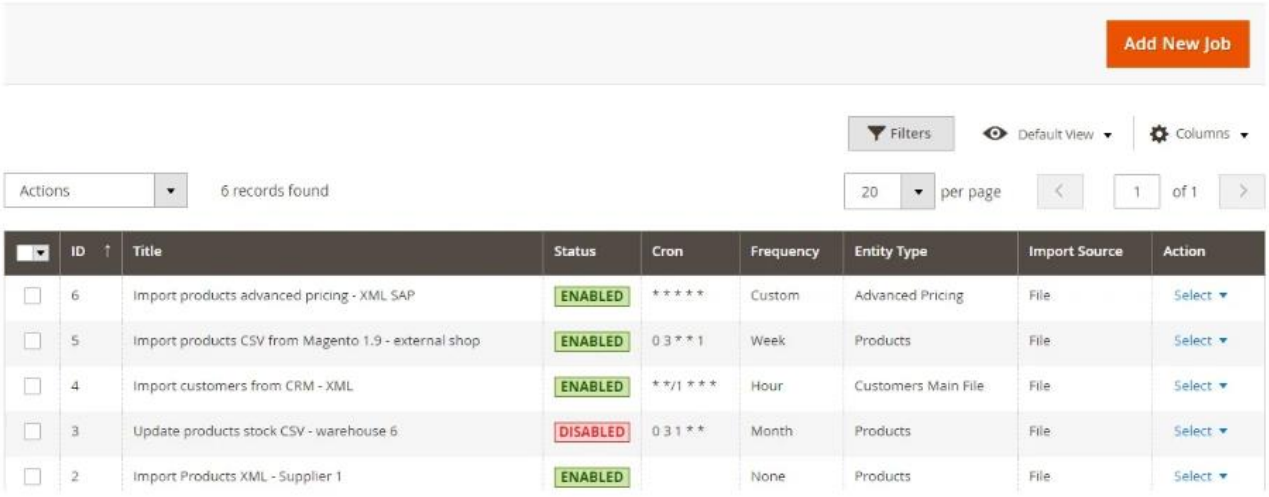


Рисунок 3.11 – Запланований імпорт

– сповіщення електроною поштою – модуль підтримує функцію сповіщень електронною поштою, що запускається результатами завдань імпорту та експорту. Користувач може ввімкнути сповіщення про невдалі та успішні виконання завдань, щоб отримувати інформацію про результати прямо зі своєї електронної скриньки. У розділі «Сповіщення електронною поштою» користувач може вибрати один із наявних у базі шаблонів і вказати одержувача та відправника (представлено на рисунку 3.12).

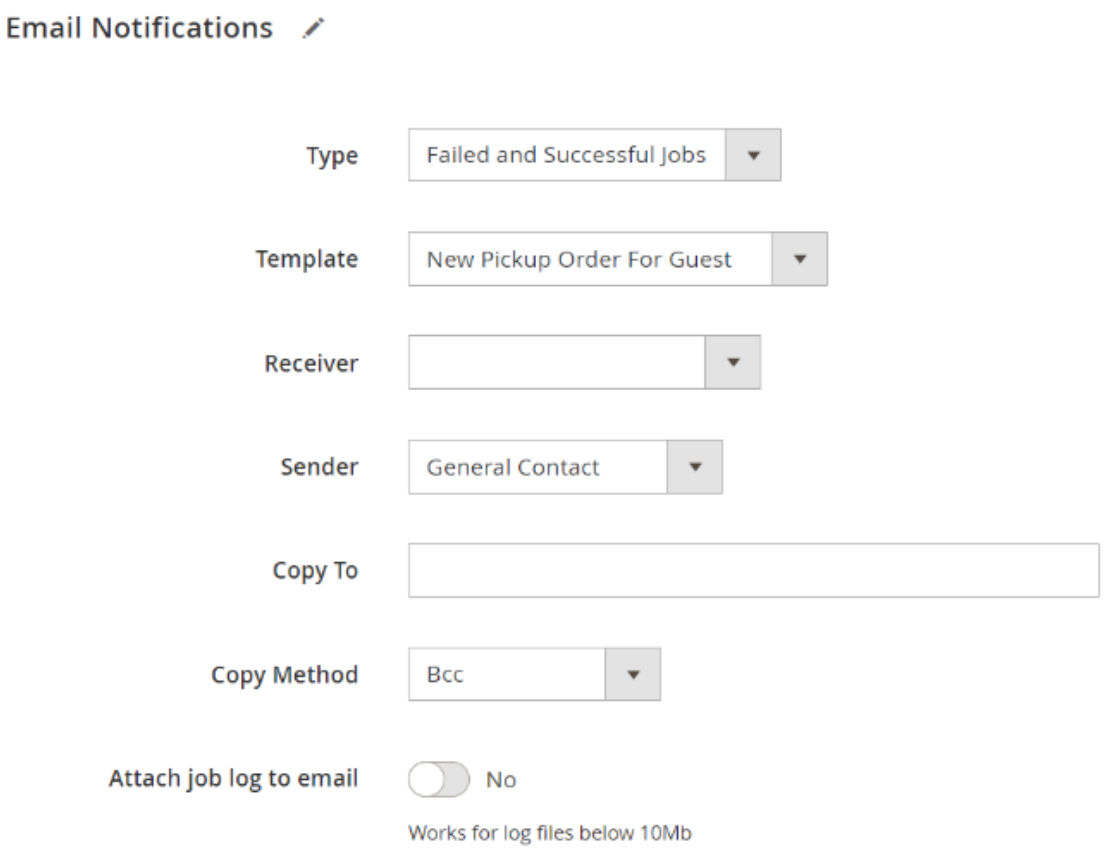


Рисунок 3.12 – Сповіщення електроною поштою

Процес імпорту:

– вибір файлу – у процесі імпорту користувач вибирає файл, що містить дані для імпорту. Модуль підтримує різні формати файлів, і ви можете налаштувати такі параметри, як роздільник і кодування файлів,

– конфігурація імпорту – далі користувач налаштовує параметри імпорту, включаючи вибір типу сутності (наприклад, продукти, клієнти, замовлення),

зіставлення стовпців файлів з атрибутами Magento 2, визначення поведінки імпорту (наприклад, створення, оновлення, видалення) і налаштування правила перевірки даних (представлено на рисунку 3.13),

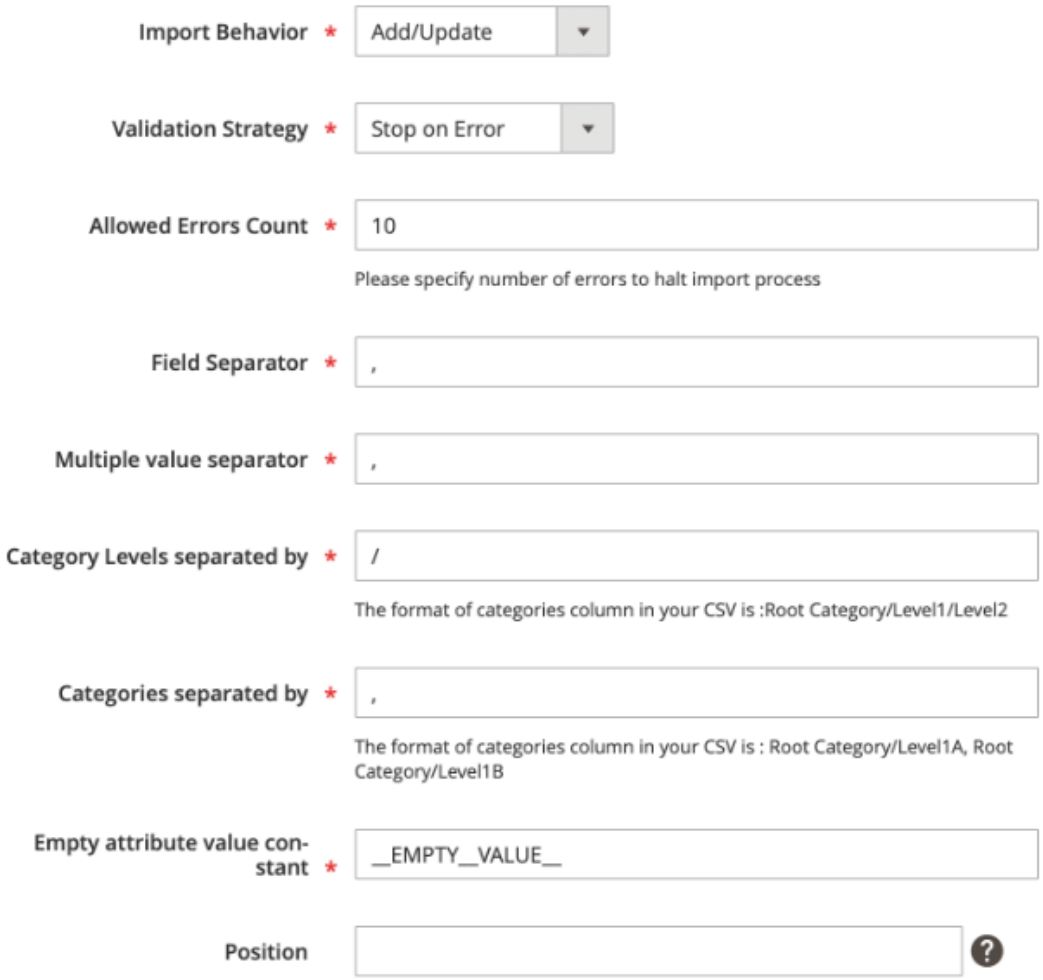


Рисунок 3.13 – Конфігурація імпорту

– перевірка даних – модуль виконує перевірку даних на основі визначених правил, щоб переконатися, що імпортовані дані відповідають необхідним стандартам. Повідомляється про помилки перевірки, і користувач може пропустити недійсні записи або зупинити процес імпорту,

– виконання імпорту – після налаштування параметрів імпорту та перевірки даних користувач починає процес імпорту. Модуль обробляє дані пакетами, оптимізуючи використання пам’яті та покращуючи продуктивність,

– результати імпорту – модуль надає зведення результатів імпорту, включаючи кількість імпортованих записів, усі виявлені помилки та час, витрачений на операцію імпорту. Користувач також може отримати доступ до детального журналу процесу імпорту для усунення несправностей і аналізу.

Процес експорту:

– вибір сутності – у процесі експорту користувач вибирає тип сутності (наприклад, продукти, клієнти, замовлення), яку потрібно експортувати (представлено на рисунку 3.14),

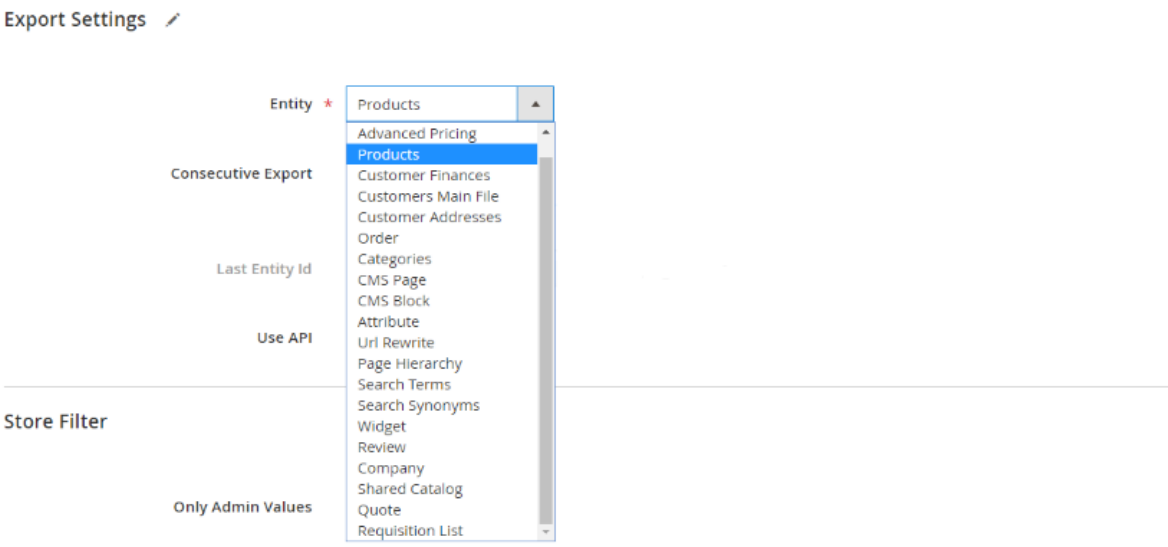


Рисунок 3.14 – Вибір сутності

– конфігурація експорту – далі користувач налаштовує параметри експорту, наприклад вибираєте формат файлу експорту, вказуєте призначення файлу та зіставляєте атрибути Magento 2 зі стовпцями файлу експорту,

– виконання експорту – після налаштування конфігурації експорту користувач починає процес експорту. Модуль отримує вибрані дані сутності з бази даних Magento 2 і генерує файл експорту у вказаному форматі,

– результати експорту – після завершення експорту модуль надає підсумок результатів експорту, включаючи кількість експортованих записів і час, витрачений на операцію експорту.

3.4 Висновки до розділу 3

Виконано програмну реалізацію технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів із визначенням особливостей.

Програмна реалізація пропонує зручний інтерфейс, інтегрований в панель адміністратора Magento 2. Інтерфейс дозволяє налаштовувати параметри імпорту/експорту, керувати профілями імпорту, відстежувати історію імпорту/експорту та виконувати різноманітні завдання керування даними. Інтерфейс забезпечує інтуїтивно зрозумілу навігацію, проводячи користувача через кроки налаштування імпорту/експорту та забезпечуючи попередній перегляд даних у реальному часі перед виконанням процесів імпорту/експорту. Користувач може отримати доступ до журналів імпорту/експорту, переглянути всі виявлені помилки та виконати завдання з усунення несправностей безпосередньо з інтерфейсу.

Виконане експериментальне тетстування реалізованої технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів, підтвердило підвищення продуктивності SQL-запиту з декомпозицією у порівнянні зі звичайним SQL-запитом на 6,6% - 9,2% в залежності від сценарію застосування.

Висновки

В кваліфікаційній роботі реалізовано технологію трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів.

Для досягнення поставленої мети було вирішено задачі аналізу предметної області, сучасних платформ електронної комерції, сучасних тенденцій покращення ефективності електронної комерції, ввиконано огляд принципів побудови запитів, реалізовано технологію трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів, виконано програмну реалізацію технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів, проведено експериментальне тестування інформаційної технології.

Результат експериментального тетстування інформаційної технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів, показало підвищення продуктивності SQL-запитів з декомпозицією у порівнянні на 6,6% - 9,2% в залежності від сценарію застосування.

Перелік посилань

1. Magento 2 Official – https://devdocs.magento.com/.
2. Shopify – http://surl.li/hgomp.
3. https://developer.bigcommerce.com/build.
4. https://woocommerce.com/docs/.
5. https://devdocs.prestashop-project.org/.
6. https://dev.mysql.com/
7. Magento 2 Optimization – https://magefan.com/blog/speed-up-magento-2.
8. Підзапити SQL - https://shorturl.at/CGHZ0.
9. Декомпозиція запитів – https://docplayer.net/27460409-Chapter-6-query-decomposition-and-data-localization.html.
10. Декомпозиція запитів та реляційна алгебра – https://www.geeksforgeeks.org/query-optimization-in-relational-algebra/
11. Оптимізація запитів SQL: як налаштувати продуктивність запитів SQL – https://blog.devart.com/how-to-optimize-sql-query.html
12. Загальні методи оптимізації запитів – https://experiencestack.co/11-sql-query-optimization-techniques-commonly-used-in-projects-ed45c31c45cd
13. Методи оптимізації запитів – https://shorturl.at/dDEL3
14. Вступ до обробки та оптимізації запитів – https://clas.iusb.edu/computer-science-informatics/research/reports/TR-20080105-1.pdf
15. Jeffrey Ullman Database Systems: The Complete Book / Гектор Гарсія-Моліна, Джеффрі Ульман і Дженніфер Відом – “Pearson”, 2003.
16. S. Venkataraman, T. Zhang. Heterogeneous Database Query Optimization in DB2 – Міжн. конф. на дуже великих базах даних, серпень 1998.
17. S. Parekh, K. Rose, J. Hellerstein, S. Lightstone, M. Hurras, V. Chang, Managing the Performance Impact of Administrative Utilities, – 2003.
18. N. Kabra, D. J. DeWitt. Efficient Mid-Query Re-Optimization of Sub-Optimal
19. Query Execution Plans – Міжн. конф. з управління с даними (Сіетл, США), червень 1998 р.

20. H. Boughton, P. Martin, W. Powley, and R. Horman. Workload Class Importance Policy in Autonomic Database Management Systems, – Семінар по політиці для розподілених систем і мереж (Лондон) 2006.
21. Andreas Lübcke, Veit Köppen, and Gunter Saake, A Query Decomposition Approach for Relational DBMS using Different Storage Architectures – Факультет інформатики Отто-фон-Геріке-Університет Магдебурга, 2011.
22. Query Optimization in DBMS using Query Decomposition and Query Caching, Міжнародний журнал комп'ютерних наук і застосувань, квітень 2013.
23. Dr. Osmar R. Zaïane, Query Processing & Optimization – Університет Альберти, 2003.
24. Query Processing and Optimization – https://www.brainkart.com/article/Query-Processing-and-Optimization-(QPO)_11372/
25. Andrey Gubichev, Query Processing and Optimization in Graph Databases – Технічний університет Мюнхена Факультет інформатики Lehrstuhl III – Datenbanksysteme, 2015.

ДОДАТКИ

Додаток А

Презентаційний матеріал

Технологія трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів

Кваліфікаційна робота бакалавра, студент Андросюк Іван **Олександрович**

Платформа електронної комерції Magento 2 e-commerce

Magento 2 — це популярна платформа електронної комерції, яка надає надійний набір функцій для створення та керування онлайн-магазинами. Ось деякі з ключових функцій і переваг Magento 2 як платформи електронної комерції, що пропонує потужну основу для створення та керування онлайн-магазином, надаючи змогу створити багатофункціональний, масштабований та персоналізований досвід електронної комерції:

Сучасні платформи електронної комерції

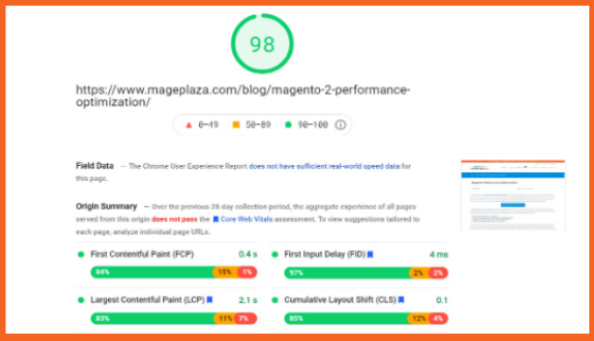
WooCommerce — це популярна платформа електронної комерції, яка функціонує як плагін для WordPress

Shopify — це комплексна платформа електронної комерції, яка надає комплексне рішення для створення та керування онлайн-магазинами

BigCommerce - це хмарна платформа електронної комерції, яка надає повністю розміщене рішення для створення онлайн-магазинів і керування ними

Сучасні тенденції покращення ефективності електронної комерції

- Оптимізація кешування
- Оптимізація продуктивності бази даних
- Оптимізація коду та конфігурації
- Конфігурація та оптимізація сервера
- Розширення та оптимізація модулів



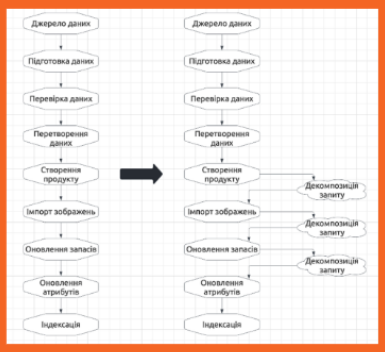
Це лише деякі з методів підвищення ефективності Magento 2. Важливо проаналізувати та оптимізувати конфігурацію, інфраструктуру та налаштування конкретного магазину на основі результатів тестування продуктивності та моніторингуЦе лише деякі з методів підвищення ефективності Magento 2.

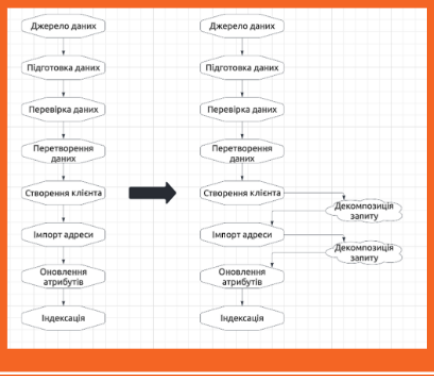
Принцип побудови технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів

Декомпозиція запиту стосується процесу розбиття складного запиту на менші прості підзапити, які можна виконувати окремо або паралельно для досягнення бажаного результату. Розбитий запит ділиться на кілька підзапитів, кожен з яких націлений на певний аспект даних у Magento 2. Наприклад, підзапити можуть зосереджуватися на отриманні деталей продукту, інформації про клієнта або даних про замовлення. Після виконання підзапитів отримані дані агрегуються в уніфікований результат. Цей крок поєднує та структурує дані відповідно до вимог вихідного запиту

Принцип побудови технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів

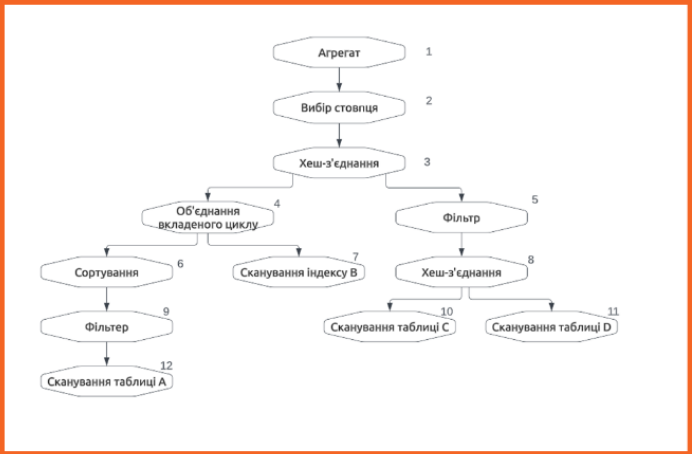
Найнеобхідніші процеси імпорту для оптимізації та збільшення продуктивності системи: імпорти продуктів, імпорти клієнтів, імпорти атрибутів та наборів атрибутів, імпорти замовлень, імпорти запасів та інвентарю.





Структура запитів з використанням принципу декомпозиції

На рисунку показано приклад QEP (планом виконання запиту планом виконання запиту). У цьому QEP дані з чотирьох різних таблиць бази даних (таблиць A, B, C і D) витягуються, фільтруються, об'єднуються, а потім агрегуються для створення бажаних кінцевих результатів. Важливо зауважити, що структура плану, показана на рисунку, є лише концептуальною структурою, а не фактичним планом від оптимізатора запитів. Він використовується лише для ілюстрації.



Особливості використання декомпозиції запитів

Опис псевдокоду першого проходу алгоритму декомпозиції показано на рисунку зправа. Він включає ітераційну процедуру під назвою «FindSegments», яка застосовується до QEP запиту. Після виконання ітераційного процесу цієї процедури всі сегменти в QEP, а також їхні зв'язки залежностей ідентифікуються та зберігаються у двох глобальних наборах, GSegSet і GSegRelSet.

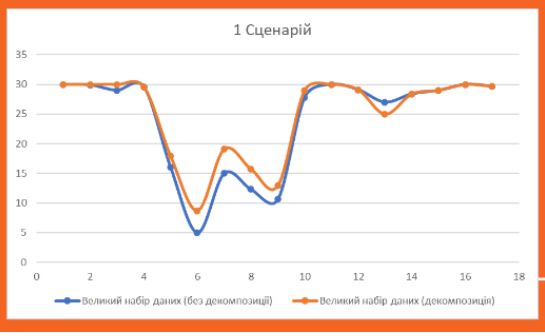
```
1   ReOrganizeSegments (QEP) {
2       call procedure "FindSegments(QEP)" to generate the global segment set "GSegSet" and the
            global segment relationship set "GSegRelSet";

3       FOR each segment(curSeg) in GSegSet
4           calculate the cost for curSeg based on QEP compiler information;
5       ENDFOR;

6       curSKF ← calculate the "skew factor" for GSegSet;
7       validSKFRange ← a pre-defined acceptable SKF range;

8       REPEAT
9           minSeg ← find the smallest segment (having minimum cost) in GSegSet ;
10          conSeg ← find the smallest segment that is connected to minSeg in GSegRelSet (either
                depends on or is depended on minSeg);
11          newCBSeg ← merge minSeg and conSeg to create a new larger CB-segment;
12          update GSegSet such that minSeg and conSeg are removed and newCBSeg is added;
13          update GSegRelSet such that all segment dependency relationships that involve minSeg
                and conSeg are modified correctly to involve newCBSeg instead;
14          curSKF ← calculate the "skew factor" for GSegSet;
15      UNTIL curSKF is within validSKFRange OR there is only one segment left in GSegSet;

16      IF there is only one segment left in GSegSet THEN
17          notify "Exception Management" module that a cost balanced solution is impossible
18      ENDIF;
19  }
```

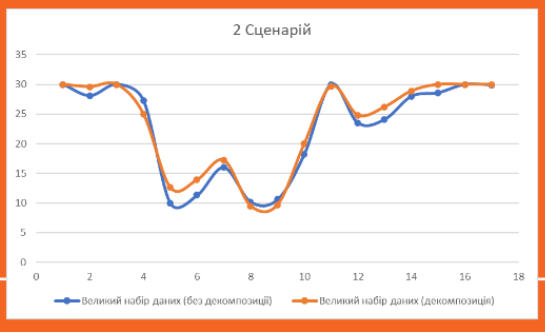

```
1   FindSegments (QEP) {
2       subTreeSet ← create a empty tree set;

3       FOR each  leaf  node (curLeaf) of QEP
4           curPath ← identifies the tree path that curLeaf belongs to in QEP;
5           cbaNode ← find its closet blocking ancestor node or the return node along curPath;
6           subTree ← create a sub-tree that is rooted at cbaNode and includes all nodes in the
                curPath from cbaNode to curLeaf;
7           add subTree  into subTreeSet;
8       ENDFOR;

9       REPEAT
10          curSubTree ← get next sub-tree from subTreeSet;
11          newSeg ← NULL;

12          IF the root node of curSubTree has only one input in the original QEP THEN
13              newSeg ←  create a new segment  that has the same structure as curSubTree;
14          ELSE
15              matchedSubTrees ← find other sub-trees that have the same root as curSubTree;
16              IF matchedSubTrees is not empty THEN
17                  newSeg ← create a new segment by merging matchedSubTrees with curSubTree
                        such that shared nodes only appear once in the segment;
18                  remove matchedSubTrees from subTreeSet;
19              ENDIF;
20          ENDIF;

21          IF newSeg is not NULL THEN
22              add newSeg into GSegSet;
23              update QEP such that the whole newSeg sub-tree in QEP is replaced by a newly
                    created virtual node;
24          ENDIF;

25          remove curSubTree from subTreeSet
26      UNTIL subTreeSet is empty;

27      REPEAT
28          call procedure "FindSegments(newQEP)";
29          segRels ← create segment dependency relationships between any two segments that are
                found in two consecutive iterations if they are connected by a virtual node;
30          add segRels into GSegRelSet;
31      UNTIL all nodes in the original QEP are processed
32  }
```

Псевдокод на рисунок зліва визначає, як досягається збалансоване за вартістю рішення шляхом об'єднання сегментів, знайдених під час першого проходу алгоритму. Він не враховує сегменти, що розкладаються. Використовуючи алгоритм, розкладання великого сегмента завжди переривається конвеєрну операцію.

Результати декомпозиції запитів при реалізації технології трансферу даних для платформи Magento 2 e-commerce

Під час проведення тестувань продуктивності з використанням декомпозиції зібрано наступні результати відображенні на рисунках нижче, де вертикальна шкала – пропускна здатність (запит/секунду), а горизонтальна шкала інтервали часу. Підрезюмувавши графіки, отримані результату дають 9.2% у сценарії 1 та 6.6% у другому сценарії, збільшення продуктивності під час виконання великих запитів. Незалежно від 1 чи 2 сценарію, у більшості випадків результатом декомпозиції запитів є збільшення пропускної здатності, тим самим пришвидшення виконання маніпуляцій з базою даних





Результати декомпозиції запитів при реалізації технології трансферу даних для платформи Magento 2 e-commerce

В кваліфікаційній роботі реалізовано технологію трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів. Для досягнення поставленої мети було вирішено задачі аналізу предметної області, сучасних платформ електронної комерції, сучасних тенденцій покращення ефективності електронної комерції, ввиконано огляд принципів побудови запитів, реалізувовано технологію трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів, виконано програмну реалізацію технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів, проведено експериментальне тестування інформаційної технології. Результат експериментального тетстування інформаційної технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів, показало підвищення продуктивності SQL-запитів з декомпозицією у порівнянні на 6,6% - 9,2% в залежності від сценарію застосування.

Додаток Б

(програмний код)

```
<?php

namespace Ivan\ImportExport\Model\Import;

use Ivan\ImportExport\Traits\Import\Entity as ImportTrait;
use Magento\Catalog\Api\CategoryRepositoryInterface;
use Magento\Catalog\Model\Category as MagentoCategoryModel;
use Magento\Catalog\Model\CategoryFactory;
use Magento\Catalog\Model\ResourceModel\Category\CollectionFactory;
use
Magento\CatalogImportExport\Model\Import\Product\CategoryProcessor;
use Magento\Cms\Model\Page\DomValidationState;
use Magento\Eav\Model\Config;
use Magento\Framework\App\ObjectManager;
use Magento\Framework\App\ResourceConnection;
use Magento\Framework\Event\ManagerInterface;
use Magento\Framework\Json\Helper\Data;
use Magento\Framework\Registry;
use Magento\Framework\Stdlib\StringUtils;
use Magento\ImportExport\Model\Import;
use Magento\ImportExport\Model\Import\Entity\AbstractEntity;
use
Magento\ImportExport\Model\Import\ErrorProcessing\ProcessingError;
use
Magento\ImportExport\Model\Import\ErrorProcessing\ProcessingErrorAggre
gatorInterface;
use Magento\ImportExport\Model\ResourceModel\Helper;
use Symfony\Component\Console\Output\ConsoleOutput;

/**
 * Class Category
 *
 * @package Ivan\ImportExport\Model\Import
 */
class Category extends AbstractEntity
{
    use ImportTrait;

    /**
     * Delimiter in category path.
     */
    const DELIMITER_CATEGORY = '/';

    const PSEUDO_MULTI_LINE_SEPARATOR = '|';

    const PAIR_NAME_VALUE_SEPARATOR = '=';

    /**
     * Column category url key.
     */
    const COL_URL = 'url_key';

    const COL_STORE = 'store_view';

    const COL_STORE_NAME = 'store_name';

    const COL_URL_PATH = 'url_path';

    /**
     * Column category name.
     */
    const COL_NAME = 'name';

    /**
     * Column category parent id.
     */
    const COL_PARENT = 'parent_id';

    /**
     * Column category path.
     */
    const COL_PATH = 'path';

    /**
     * Column is active.
     */
    const COL_IS_ACTIVE = 'is_active';

    /**
     * Column Include in Menu.
     */
    const COL_INCLUDE_IN_MENU = 'include_in_menu';

    /**
     * Column Custom layout update.
     */
    const COL_CUSTOM_LAYOUT_UPDATE = 'custom_layout_update';

    /**
     * Error codes
     */
    const ERROR_CODE_NAME_REQUIRED = 'columnNameIsRequired';
    const ERROR_CODE_LAYOUT_UPDATE_IS_NOT_VALID =
'CustomLayoutIsNotValid';

    protected $errorTemplates = [
        self::ERROR_CODE_NAME_REQUIRED => "Column 'name' is not set",
        self::ERROR_CODE_LAYOUT_UPDATE_IS_NOT_VALID => "Column
'custom_layout_update' is not valid"
    ];

    /**
     * Core event manager proxy
     *
     * @var ManagerInterface
     */
    protected $eventManager = null;

    /**
     * Flag for replace operation.
     *
     * @var null
     */
    protected $replaceFlag = null;

    /**
     * @var CategoryProcessor
     */
    protected $categoryProcessor;

    /**
     * @var CollectionFactory
     */
    protected $categoryColFactory;

    /**
     * @var CategoryFactory
     */
    protected $categoryFactory;

    protected $storeManager;

    /**
     * @var CategoryRepositoryInterface
     */
    protected $categoryRepository;

    protected $resource;

    protected $resourceFactory;

    /**
     * @var Registry
     */
    protected $registry;

    /**
     * Categories text-path to ID hash.
     *
     * @var array
     */
    protected $categories = [];

    /**
     * @var array
     */
    protected $categoriesCache = [];

    protected $categoriesUrl;

    /**
     * @var \Magento\Framework\Filter\FilterManager
     */
    protected $filterManager;

    /**
     * @var DomValidationState
     */
    private $validationState;

    private $multiLineSeparatorForRegexp;

    protected $attributeCache = [];

    protected $attrData = [];

    protected $attributeCol;

    protected $sourceType;

    protected $nameToId;

    /**
     * @var \Magento\Framework\App\ProductMetadata
     */
    public $productMetadata;

    /**
     * @var \Ivan\ImportExport\Helper\Additional
     */
```

```
    protected $additional;

    /**
     * @var
\Magento\Framework\View\Model\Layout\Update\ValidatorFactory
     */
    protected $validatorFactory;

    /**
     * @var array
     */
    protected $customAttr = [
        'custom_apply_to_products',
        'custom_design',
        'custom_design_from',
        'custom_design_to',
        'custom_layout_update',
        'custom_use_parent_settings',
        'description'
    ];

    /**
     * @var array
     */
    protected $categoriesDeleted = [];

    /**
     * @var array
     */
    protected $urlComparableList = [];

    /**
     * @var array
     */
    protected $urlRequestPathStoreId = [];

    /**
     * @var array
     */
    protected $foundDuplicate = [];

    /**
     * @param Data $jsonHelper
     * @param \Magento\ImportExport\Helper\Data $importExportData
     * @param \Magento\ImportExport\Model\ResourceModel\Import\Data
$importData
     * @param Config $config
     * @param ResourceConnection $resource
     * @param Helper $resourceHelper
     * @param StringUtils $string
     * @param ProcessingErrorAggregatorInterface $errorAggregator
     * @param CollectionFactory $categoryColFactory
     * @param CategoryProcessor $categoryProcessor
     * @param CategoryFactory $categoryFactory
     * @param ManagerInterface $eventManager
     * @param \Magento\Store\Model\StoreManagerInterface
$storeManager
     * @param CategoryRepositoryInterface $categoryRepository
     * @param ConsoleOutput $output
     * @param \Magento\Framework\Registry $registry
     * @param \Ivan\ImportExport\Model\ResourceModel\Import\Data
$importFireData
     * @param
\Magento\Catalog\Model\ResourceModel\Category\Attribute\CollectionFact
ory $attributeColFactory
     * @param \Magento\Catalog\Model\ResourceModel\CategoryFactory
$categoryResourceFactory
     * @param \Ivan\ImportExport\Helper\Additional $additional
     * @param \Magento\Framework\App\ProductMetadata $productMetadata
     * @param
\Magento\Framework\View\Model\Layout\Update\ValidatorFactory
$validatorFactory
     * @param \Magento\Framework\Filter\FilterManager $filterManager
     * @param DomValidationState $validationState
     *
     * @throws \Magento\Framework\Exception\LocalizedException
     */
    public function __construct(
        Data $jsonHelper,
        \Magento\ImportExport\Helper\Data $importExportData,
        \Magento\ImportExport\Model\ResourceModel\Import\Data
$importData,
        Config $config,
        ResourceConnection $resource,
        Helper $resourceHelper,
        StringUtils $string,
        ProcessingErrorAggregatorInterface $errorAggregator,
        CollectionFactory $categoryColFactory,
        CategoryProcessor $categoryProcessor,
        CategoryFactory $categoryFactory,
        ManagerInterface $eventManager,
        \Magento\Store\Model\StoreManagerInterface $storeManager,
        CategoryRepositoryInterface $categoryRepository,
        ConsoleOutput $output,
        Registry $registry,
        \Ivan\ImportExport\Model\ResourceModel\Import\Data
$importFireData,
        \Magento\Catalog\Model\ResourceModel\Category\Attribute\Colle
ctionFactory $attributeColFactory,
        \Magento\Catalog\Model\ResourceModel\CategoryFactory
$categoryResourceFactory,
        \Ivan\ImportExport\Helper\Additional $additional,
        \Magento\Framework\App\ProductMetadata $productMetadata,
        \Magento\Framework\View\Model\Layout\Update\ValidatorFactory
$validatorFactory,
        \Magento\Framework\Filter\FilterManager $filterManager,
        $validationState = null
    ) {
        $this->categoryColFactory = $categoryColFactory;
        $this->categoryProcessor = $categoryProcessor;
        $this->categoryFactory = $categoryFactory;
        $this->categoryRepository = $categoryRepository;
        $this->eventManager = $eventManager;
        $this->registry = $registry;
        $this->storeManager = $storeManager;
        $this->output = $output;
        $this->attributeCol = $attributeColFactory;
        $this->resourceFactory = $categoryResourceFactory;
        $this->additional = $additional;
        $this->productMetadata = $productMetadata;
        $this->validatorFactory = $validatorFactory;
        $this->validationState = $validationState;

        if (version_compare($this->productMetadata->getVersion(),
'2.2.2', '>=') && !$validationState) {
            $this->validationState = ObjectManager::getInstance()-
>get(
                DomValidationState::class
            );
        }
        parent::__construct(
            $jsonHelper,
            $importExportData,
            $importData,
            $config,
            $resource,
            $resourceHelper,
            $string,
            $errorAggregator
        );
        $this->_dataSourceModel = $importFireData;
        $this->initCategories()->initAttributes();
        $this->initRequestPathStoreId();
        $this->filterManager = $filterManager;
    }

    /**
     * Prepare all existing categories in array
     * @return $this
     * @throws \Magento\Framework\Exception\LocalizedException
     */
    protected function initCategories()
    {
        if (empty($this->categories)) {
            $stores = $this->storeManager->getStores();
            $searchStores =
[\Magento\Store\Model\Store::DEFAULT_STORE_ID];
            $this->nameToId['admin'] =
\Magento\Store\Model\Store::DEFAULT_STORE_ID;
            foreach ($stores as $store) {
                $this->nameToId[$store->getCode()] = $store->getId();
                $searchStores[] = $store->getId();
            }
            foreach ($searchStores as $store) {
                $collection = $this->categoryColFactory->create();
                $collection->setStoreId($store)
                    ->addAttributeToSelect(self::COL_NAME)
                    ->addAttributeToSelect(self::COL_URL);
                /** @var \Magento\Catalog\Model\Category $category */
                foreach ($collection as $category) {
                    $structure = explode(self::DELIMITER_CATEGORY,
$category->getPath());
                    $pathSize = count($structure);
                    $this->categoriesCache[$category->getId()] =
$category;

                    if ($pathSize > 1) {
                        $path = [];
                        for ($i = 1; $i < $pathSize; $i++) {
                            $path[] = $collection-
>getItemById((int)$structure[$i])->getName();
                        }
                        $index = implode(self::DELIMITER_CATEGORY,
$path);
                        $this->categories[$index] = $category-
>getId();
                    } else {
                        $this->categories[$category->getName()] =
$category->getId();
                    }
                }
            }
        }

        $this->setupKeyUrls();

        return $this;
    }

    protected function initAttributes()
    {
        foreach ($this->attributeCol->create() as $item) {
            $this->attrData[$item->getAttributeCOde()] = $item-
>getData();
        }
    }

    protected function searchInCategories($id, $data)
    {
        $array = [];
```

```
        foreach ($data as $el) {
            if ($el['entity_id'] == $id) {
                $array[$el['value']];
            }
        }

        return $array;
    }

    /**
     * Create Category entity from raw data.
     *
     * @throws \Exception
     * @return bool Result of operation.
     * @SuppressWarnings(PHPMD.CyclomaticComplexity)
     */
    protected function _importData()
    {
        /**
         * Add templates here because we rewrite aggregator in
General Trait
         */
        foreach ($this->errorTemplates as $errorCode => $message) {
            $this->errorAggregator
                ->addErrorMessageTemplate($errorCode, $message);
        }

        $this->_validatedRows = null;
        if (Import::BEHAVIOR_DELETE == $this->getBehavior()) {
            $this->deleteCategories();
        } elseif (Import::BEHAVIOR_REPLACE == $this->getBehavior()) {
            $this->replaceProcess();
        } else {
            $this->saveCategoriesData();
        }
        $this->eventManager-
>dispatch('catalog_category_import_finish_before', ['adapter' =>
$this]);

        return true;
    }

    /**
     * Delete categories is delete behavior is selected
     * @return $this
     * @throws \Magento\Framework\Exception\InputException
     * @throws \Magento\Framework\Exception\NoSuchEntityException
     */
    protected function deleteCategories()
    {
        $categoryId = null;
        while ($bunch = $this->_dataSourceModel->getNextBunch()) {
            $this->categoriesCache = [];
            foreach ($bunch as $rowNum => $rowData) {
                if (!$this->validateRow($rowData, $rowNum)) {
                    continue;
                }
                if ($this->getErrorAggregator()->hasToBeTerminated())
{
                    $this->getErrorAggregator()-
>addRowToSkip($rowNum);
                    continue;
                }
                if (isset($rowData['name']) && isset($this-
>categories[$rowData['name']])) {
                    $categoryId = (int)$this-
>categories[$rowData['name']];
                } elseif (isset($rowData['entity_id'])) {
                    $categoryId = (int)$rowData['entity_id'];
                }

                if ($categoryId) {
                    if ($this->categoryFactory->create()->
                    getCollection()->addFieldToFilter('entity_id',
$categoryId)
                        ->getSize()) {
                        try {
                            $category = $this->categoryRepository-
>get($categoryId);
                            if ($this->getResource()-
>isForbiddenToDelete($categoryId)) {
                                $this->addRowError(
                                    'Cannot delete category ',
                                    $rowNum
                                );
                            } else {
                                $this->categoryRepository-
>delete($category);
                                if (Import::BEHAVIOR_REPLACE ==
$this->getBehavior()) {
                                    if (isset($rowData['name'])) {
                                        $this->categoriesDeleted[] =
$rowData['name'];
                                    }
                                }
                            }
                        } catch
(\Magento\Framework\Exception\StateException $e) {
                            $this->addRowError(
                                $e->getMessage(),
                                $rowNum
                            );
                        }
                    }
                } else {
                    $this->addRowError(
                        'Cannot delete category ',
                        $rowNum
                    );
                }
            }
        }

        return $this;
    }

    /**
     * Delete all categories when replace behavior is selected
     * @return $this
     * @throws \Magento\Framework\Exception\InputException
     * @throws \Magento\Framework\Exception\LocalizedException
     * @throws \Magento\Framework\Exception\NoSuchEntityException
     */
    protected function deleteAllCategories()
    {
        $this->deleteCategories();

        /**
         * Clear categories cache.
         */
        $this->categories = [];
        $this->categoriesCache = [];

        /**
         * Re-init default categories.
         */
        $this->initCategories();

        return $this;
    }

    /**
     * @return $this
     * @throws \Magento\Framework\Exception\InputException
     * @throws \Magento\Framework\Exception\LocalizedException
     * @throws \Magento\Framework\Exception\NoSuchEntityException
     */
    protected function replaceProcess()
    {
        $this->deleteAllCategories();
        $this->saveCategoriesData();

        return $this;
    }

    /**
     * Gather and save information about product entities.
     *
     * @return $this
     * @SuppressWarnings(PHPMD.CyclomaticComplexity)
     * @SuppressWarnings(PHPMD.NPathComplexity)
     * @SuppressWarnings(PHPMD.ExcessiveMethodLength)
     * @SuppressWarnings(PHPMD.UnusedLocalVariable)
     */
    protected function saveCategoriesData()
    {
        $this->_initSourceType('url');
        $groupCategoryId = [];
        while ($bunch = $this->_dataSourceModel->getNextBunch()) {
            $in = 0;
            $up = 0;
            $this->categoriesCache = [];
            $bunch = $this->prepareImagesFromSource($bunch);
            foreach ($bunch as $rowNum => $rowData) {
                if ($rowData['name'] == 'Root Catalog') {
                    continue;
                }
                $this->_processedRowsCount++;
                $rowData = $this->joinIdenticalyData($rowData);
                $rowData = $this->customChangeData($rowData);
                $rowData = $this->clearEmptyData($rowData, $rowNum);

                if (!$rowData) {
                    continue;
                }

                if (!isset($rowData[self::COL_NAME])) {
                    $this->getErrorAggregator()->addError(
                        self::ERROR_CODE_NAME_REQUIRED,
                        ProcessingError::ERROR_LEVEL_CRITICAL,
                        $this->_processedRowsCount
                    );
                    continue;
                }

                if (isset($rowData[self::COL_CUSTOM_LAYOUT_UPDATE])
                    &&
!empty($rowData[self::COL_CUSTOM_LAYOUT_UPDATE])) {
                    $rowData[self::COL_CUSTOM_LAYOUT_UPDATE] =
stripslashes($rowData[self::COL_CUSTOM_LAYOUT_UPDATE]);
                    if (!$this-
>validateLayoutUpdateRow($rowData[self::COL_CUSTOM_LAYOUT_UPDATE])) {
                        $this->getErrorAggregator()->addError(
                            self::ERROR_CODE_LAYOUT_UPDATE_IS_NOT_VAL
ID,

                            ProcessingError::ERROR_LEVEL_WARNING,
                            $this->_processedRowsCount
                        );
                        continue;
                    }
```

```
                }

                if (!$this->validateRow($rowData, $rowNum)) {
                    $this->addLogWriteln(
                        __('Category with name: %1 is not valided',
$rowData[self::COL_NAME]),
                        $this->output,
                        'info'
                    );
                    continue;
                }
                $time = explode(" ", microtime());
                $startTime = $time[0] + $time[1];
                $name = $rowData[self::COL_NAME];

                if ($this->getErrorAggregator()->hasToBeTerminated())
{
                    $this->getErrorAggregator()-
>addRowToSkip($rowNum);
                    continue;
                }

                if (Import::BEHAVIOR_REPLACE == $this->getBehavior())
{
                    if (isset($rowData['entity_id'])) {
                        unset($rowData['entity_id']);
                    }
                    if (!empty($this->categoriesDeleted)) {
                        if (in_array($rowData[self::COL_NAME], $this-
>categoriesDeleted)) {
                            $rowData[self::COL_NAME];
                        } else {
                            continue;
                        }
                    }
                }

                $rowData = $this->changeData($rowData);
                $rowData['store_id'] = 0;
                if (!empty($rowData[self::COL_STORE])) {
                    if (isset($this-
>nameToId[$rowData[self::COL_STORE]])) {
                        $rowData['store_id'] = $this-
>nameToId[$rowData[self::COL_STORE]];
                        unset($rowData[self::COL_STORE]);
                    } else {
                        $this->addRowError(
                            "Store could not find for this
category:".$rowData[self::COL_NAME],
                            $this->_processedRowsCount
                        );
                    }
                }
                $rowPath = $rowData[self::COL_NAME];
                if (!empty($rowPath)) {
                    if (is_int($rowPath)) {
                        try {
                            /** @var \Magento\Catalog\Model\Category
$category */
                            $category = $this->categoryFactory-
>create();
                            if (!($parentCategory = isset($this-
>categoriesCache[$rowPath])
                                ? $this->categoriesCache[$rowPath] :
null)
                            ) {
                                $parentCategory = $this-
>categoryFactory->create()->load($rowPath);
                            }
                            $category->setParentId($rowPath);
                            $category->setIsActive(
                                isset($rowData[self::COL_IS_ACTIVE])
? $rowData[self::COL_IS_ACTIVE] : true
                            );
                            $category->setIncludeInMenu(
                                isset($rowData[self::COL_INCLUDE_IN_M
ENU]) ? $rowData[self::COL_INCLUDE_IN_MENU] : true
                            );
                            $category->setAttributeSetId($category-
>getDefaultAttributeSetId());
                            $category-
>setStoreId($rowData['store_id']);
                            $category->addData($rowData);

                            $category->setPath($parentCategory-
>getPath());
                            $category->save();
                            $this->categoriesCache[$category-
>getId()] = $category;
                            $in++;
                        } catch (\Exception $e) {
                            $this->getErrorAggregator()->addError(
                                $e->getCode(),
                                ProcessingError::ERROR_LEVEL_NOT_CRIT
ICAL,
                                $this->_processedRowsCount,
                                null,
                                $e->getMessage()
                            );
                        }
                    } else {
                        $rowPathWithDefaultDelimiter = str_replace(
                            $this-
>_parameters['category_levels_separator'],
                            self::DELIMITER_CATEGORY,
                            $rowPath
                        );

                        if (isset($rowData['group']) &&
!empty($rowData['group'])) {
                            $catname = $rowData['_actual_name'];
                            $catid = '';
                            if
(isset($groupCategoryId[$rowData['group']])
                                &&
!empty($groupCategoryId[$rowData['group']])) {
                                $catid =
$groupCategoryId[$rowData['group']];
                            }
                            if (!$catid) {
                                $catid = isset($this-
>categories[$rowPath])
                                ? $this->categories[$rowPath] :'';
                            }

                            if ($catid) {
                                //update
                                ++$up;
                                $result = $this-
>updateCategoriesByPath($rowPathWithDefaultDelimiter, $rowData,
$catid);
                                $result = true;
                                $groupCategoryId[$rowData['group']] =
$catid;
                            } else {
                                //insert
                                ++$in;
                                $result = $this-
>prepareCategoriesByPath($rowPath, $rowData);
                                $groupCategoryId[$rowData['group']] =
$result;
                            }
                        } elseif (isset($rowData['entity_id']) &&
!empty($rowData['entity_id'])) {
                            ++$up;
                            $result = $this->updateCategoriesByPath(
                                $rowPathWithDefaultDelimiter,
                                $rowData,
                                $rowData['entity_id']
                            );
                        } elseif (!isset($this-
>categories[$rowPathWithDefaultDelimiter])) {
                            ++$in;
                            $result = $this-
>prepareCategoriesByPath($rowPath, $rowData);
                        } else {
                            ++$up;
                            $result = $this-
>updateCategoriesByPath($rowPathWithDefaultDelimiter, $rowData);
                        }
                        if ($result === false) {
                            continue;
                        }
                    }
                }
                $time = explode(" ", microtime());
                $endTime = $time[0] + $time[1];
                $totalTime = $endTime - $startTime;
                $totalTime = round($totalTime, 5);
                $this->addLogWriteln(__('category with name: %1 ....
%2s', $name, $totalTime), $this->output, 'info');
            }
            $this->addLogWriteln(__('Imported: %1 rows', $in), $this-
>output, 'info');
            $this->addLogWriteln(__('Updated: %1 rows', $up), $this-
>output, 'info');

            $this->eventManager->dispatch(
                'catalog_category_import_bunch_save_after',
                ['adapter' => $this, 'bunch' => $bunch]
            );
        }
        return $this;
    }

    /**
     * @param $rowData
     * @param $rownum
     * @return array
     */
    protected function clearEmptyData($rowData, $rownum)
    {
        foreach ($this->attrData as $attrDatum) {
            if (isset($rowData[$attrDatum['attribute_code']]) &&
$rowData[$attrDatum['attribute_code']] == '') {
                if ($attrDatum['is_required'] &&
                    !in_array($attrDatum['attribute_code'],
['available_sort_by','default_sort_by'])) {
                    $message = __('A required attribute missing %1',
$attrDatum['attribute_code']);
                    $this->getErrorAggregator()->addError(
                        self::ERROR_CODE_NAME_REQUIRED,
                        ProcessingError::ERROR_LEVEL_CRITICAL,
                        $rownum,
                        $attrDatum['attribute_code'],
                        $message,
                        $message
                    );
                    return [];
                } else {
```

```
                    unset($rowData[$attrDatum['attribute_code']]);
                }
            }
        }
        return $rowData;
    }

    /**
     * Prepare new category by path.
     *
     * @param $rowPath
     * @param $rowData
     *
     * @return bool
     */
    protected function prepareCategoriesByPath($rowPath, $rowData)
    {
        $result = true;
        $parentId = MagentoCategoryModel::TREE_ROOT_ID;
        $pathParts = explode($this-
>_parameters['category_levels_separator'], $rowPath);
        $path = '';
        foreach ($pathParts as $pathPart) {
            if ($pathPart == '') {
                continue;
            }
            $path .= $pathPart;
            if (!isset($this->categories[$path])) {
                try {
                    $category = $this->categoryFactory->create();
                    if (!($parentCategory = isset($this-
>categoriesCache[$parentId])
                        ? $this->categoriesCache[$parentId] : null)
                    ) {
                        $parentCategory = $this->categoryFactory-
>create()->load($parentId);
                    }
                    $category->addData($rowData);
                    $category->setStoreId(0);
                    $category->setParentId($parentId);
                    $category-
>setIsActive(isset($rowData[self::COL_IS_ACTIVE]) ?
$rowData[self::COL_IS_ACTIVE] : true);
                    $category->setIncludeInMenu(
                        isset($rowData[self::COL_INCLUDE_IN_MENU]) ?
$rowData[self::COL_INCLUDE_IN_MENU] : true
                    );
                    $category->setAttributeSetId($category-
>getDefaultAttributeSetId());
                    $category->setName($pathPart);
                    if
(isset($rowData[MagentoCategoryModel::KEY_AVAILABLE_SORT_BY])
                        &&
!empty($rowData[MagentoCategoryModel::KEY_AVAILABLE_SORT_BY])
                    ) {
                        $attrValue = \explode(
                            $this->getMultipleValueSeparator(),
                            $rowData[MagentoCategoryModel::KEY_AVAILA
BLE_SORT_BY]
                        );
                        $category->setAvailableSortBy($attrValue);
                    }
                    $category->setPath($parentCategory->getPath());
                    $category->save();
                    if ($category->getId()) {
                        $category->setPath($parentCategory->getPath()
. self::DELIMITER_CATEGORY . $category->getId());
                        $category->save();
                    }
                    $this->categoriesCache[$category->getId()] =
$category;
                    $this->categories[$path] = $category->getId();
                    if (!empty($rowData[self::COL_STORE_NAME])) {
                        $this->updateCategoriesByPath($rowPath,
$rowData);
                    }
                    $result = $category->getId();
                } catch (\Exception $e) {
                    $this->getErrorAggregator()->addError(
                        $e->getCode(),
                        ProcessingError::ERROR_LEVEL_NOT_CRITICAL,
                        $this->_processedRowsCount,
                        null,
                        $e->getMessage()
                    );
                    $result = false;
                }
            }
            if (isset($this->categories[$path])) {
                $parentId = $this->categories[$path];

                $path .= self::DELIMITER_CATEGORY;
            }
        }

        return $result;
    }

    /**
     * Update existing category by path.
     *
     * @param $rowPath
     * @param $rowData
     *
     * @return bool
     */
    protected function updateCategoriesByPath($rowPath, $rowData,
$entityid = 0)
    {
        $result = true;
        if ($entityid) {
            $categoryId = $entityid;
        } else {
            $categoryId = $this->categories[$rowPath];
        }
        $category = $this->categoryFactory->create()-
>setStoreId($rowData['store_id'])->load($categoryId);
        /**
         * Avoid changing category name and path
         */
        if (isset($rowData[self::COL_STORE_NAME]) &&
!empty($rowData[self::COL_STORE_NAME])) {
            $rowData[self::COL_NAME] =
$rowData[self::COL_STORE_NAME];
            unset($rowData[self::COL_STORE_NAME]);
        } elseif (isset($rowData[self::COL_NAME])) {
            if ($entityid) {
                //update store view category name so no need to add
name
                $categoryname = isset($rowData['_actual_name'])
                ? $rowData['_actual_name'] :
$rowData[self::COL_NAME];
                $rowData[self::COL_NAME] = $categoryname;
            } else {
                unset($rowData[self::COL_NAME]);
            }
        }
        if (isset($rowData[self::COL_STORE]) &&
empty($rowData[self::COL_STORE])) {
            unset($rowData[self::COL_STORE]);
        }
        if (isset($rowData[self::COL_PATH])) {
            unset($rowData[self::COL_PATH]);
        }
        try {
            foreach (\array_keys($this->attrData) as $attrCode) {
                if (!isset($rowData[$attrCode])) {
                    continue;
                }
                if (!empty($rowData[$attrCode])
                    && $category->getData($attrCode) !==
$rowData[$attrCode]
                ) {
                    if ($attrCode ===
MagentoCategoryModel::KEY_AVAILABLE_SORT_BY) {
                        $attrValue = \explode(
                            $this->getMultipleValueSeparator(),
                            $rowData[$attrCode]
                        );
                        $category->setData($attrCode, $attrValue);
                    } else {
                        $category->setData($attrCode,
$rowData[$attrCode]);
                    }
                }
            }

            $category->setStoreId($rowData['store_id']);
            /**
             * set url_key in OrigData for method
\Magento\Framework\Model\AbstractModel::dataHasChangedFor
             * cause url_path was change
             */
            if ((Import::BEHAVIOR_APPEND == $this->getBehavior()
                && $this->_parameters['generate_url'] == 1
                && isset($rowData['is_url_path_generated'])
                && $rowData['is_url_path_generated'] == 1)
                || (Import::BEHAVIOR_REPLACE == $this->getBehavior()
                && $this->_parameters['generate_url'] == 1
                && isset($rowData['is_url_path_generated'])
                && $rowData['is_url_path_generated'] == 1)
            ) {
                $urlCheck = $rowData['url_key'] . '1';
                $category->setOrigData('url_key', $urlCheck);
            }
            $category->save();
        } catch (\Exception $e) {
            $this->getErrorAggregator()->addError(
                $e->getCode(),
                ProcessingError::ERROR_LEVEL_NOT_CRITICAL,
                $this->_processedRowsCount,
                null,
                $e->getMessage()
            );
            $result = false;
        }

        return $result;
    }

    /**
     * Validate data row.
     *
     * @param array $rowData
     * @param int $rowNum
     * @return boolean
     * @SuppressWarnings(PHPMD.CyclomaticComplexity)
     * @SuppressWarnings(PHPMD.NPathComplexity)
     * @SuppressWarnings(PHPMD.ExcessiveMethodLength)
     */
```

```
    public function validateRow(array $rowData, $rowNum)
    {
        if (isset($this->_validatedRows[$rowNum])) {
            // check that row is already validated
            return !$this->getErrorAggregator()->isRowInvalid($rowNum);
        }
        $this->_validatedRows[$rowNum] = true;
        $this->_processedEntitiesCount++;

        return !$this->getErrorAggregator()->isRowInvalid($rowNum);
    }

    /**
     * Validate custom update row.
     *
     * @param string $validateString
     * @return boolean
     */
    protected function validateLayoutUpdateRow($validateString)
    {
        $layoutXmlValidator = $this->validatorFactory->create(
            [
                'validationState' => $this->validationState,
            ]
        );
        try {
            return $layoutXmlValidator->isValid($validateString);
        } catch (\Exception $e) {
            return false;
        }
    }

    /**
     * EAV entity type code getter.
     *
     * @abstract
     * @return string
     */
    public function getEntityTypeCode()
    {
        return 'catalog_category';
    }

    protected function setupKeyUrls()
    {
        $this->categoriesUrl = [];
        $collection = $this->categoryColFactory->create();
        $collection->addAttributeToSelect(self::COL_URL);
        foreach ($collection as $category) {
            $this->categoriesUrl[] = $category[self::COL_URL];
        }
    }

    /**
     * @return array
     */
    protected function getDataAttributes()
    {
        $category = $this->categoryFactory->create()->getResource();
        $attr = $category->getAttribute(self::COL_NAME);
        $attrName = $attr->getId();
        $table = $attr->getBackendTable();
        $entityTable = $category->getEntityTable();
        $collection = $this->categoryColFactory->create();
        $connection = $collection->getConnection();
        $indexList = $connection->getIndexList($entityTable);
        $entityIdField = $indexList[$connection->getPrimaryKeyName($entityTable)]['COLUMNS_LIST'][0];
        $stores = $this->storeManager->getStores();
        $searchStores = [\Magento\Store\Model\Store::DEFAULT_STORE_ID];
        foreach ($stores as $store) {
            $searchStores[] = $store->getId();
        }
        $select = $connection->select()->from(
            ['t_d' => $table],
            [$entityIdField, 'value']
        )
            ->where(
                't_d.attribute_id=?',
                $attrName
            )
            ->where(
                't_d.store_id IN(?)',
                $searchStores
            )
            ->where(
                't_d.store_id = ?',
                $connection->getIfNullSql('t_d.store_id', \Magento\Store\Model\Store::DEFAULT_STORE_ID)
            );

        return $this->_connection->fetchAll($select);
    }

    protected function _saveValidatedBunches()
    {
        $source = $this->_getSource();
        $currentDataSize = 0;
        $bunchRows = [];
        $startNewBunch = false;
        $nextRowBackup = [];
        $maxDataSize = $this->_resourceHelper->getMaxDataSize();
        $bunchSize = $this->_importExportData->getBunchSize();

        $source->rewind();
        $this->_dataSourceModel->cleanBunches();
        $file = null;
        $jobId = null;
        if (isset($this->_parameters['file'])) {
            $file = $this->_parameters['file'];
        }
        if (isset($this->_parameters['job_id'])) {
            $jobId = $this->_parameters['job_id'];
        }
        while ($source->valid() || $bunchRows) {
            if ($startNewBunch || !$source->valid()) {
                $this->_dataSourceModel->saveBunches(
                    $this->getEntityTypeCode(),
                    $this->getBehavior(),
                    $jobId,
                    $file,
                    $bunchRows
                );
                $bunchRows = $nextRowBackup;
                $currentDataSize = strlen(\Zend\Serializer\Serializer::serialize($bunchRows));
                $startNewBunch = false;
                $nextRowBackup = [];
            }
            if ($source->valid()) {
                try {
                    $rowData = $source->current();
                } catch (\InvalidArgumentException $e) {
                    $this->addRowError($e->getMessage(), $this->_processedRowsCount);
                    $this->_processedRowsCount++;
                    $source->next();
                    continue;
                }

                $this->_processedRowsCount++;
                $rowData = $this->customBunchesData($rowData);
                $rowData = $this->customFieldsMapping($rowData, $source->key());

                if ($this->validateRow($rowData, $source->key())) {
                    $rowSize = strlen($this->jsonHelper->jsonEncode($rowData));

                    $isBunchSizeExceeded = $bunchSize > 0 && count($bunchRows) >= $bunchSize;

                    if ($currentDataSize + $rowSize >= $maxDataSize || $isBunchSizeExceeded) {
                        $startNewBunch = true;
                        $nextRowBackup = [$source->key() => $rowData];
                    } else {
                        $bunchRows[$source->key()] = $rowData;
                        $currentDataSize += $rowSize;
                    }
                }

                $source->next();
            }
        }

        return $this;
    }

    /**
     * @param $rowData
     * @param $rowNum
     * @return mixed
     */
    protected function findUrlKeyDuplicates($rowData, $rowNum)
    {
        $storeCode = isset($rowData[self::COL_STORE]) ? $rowData[self::COL_STORE] : 'default';
        $rowData = $this->urlPathSlashTrim($rowData);
        if (!isset($this->urlComparableList['url_path'][$storeCode])) {
            $this->urlComparableList['url_path'][$storeCode] = [];
        }
        if (!isset($this->urlComparableList['url_key'][$storeCode])) {
            $this->urlComparableList['url_key'][$storeCode] = [];
        }

        $rowData = $this->checkCategoryIfParentIsDuplicate($storeCode, $rowData, $rowNum);

        if (!empty($rowData[self::COL_URL_PATH])
            && count(explode('/', $rowData[self::COL_URL_PATH])) == 1
            && !empty($rowData[self::COL_URL])
        ) {
            if ($rowData[self::COL_URL] != $rowData[self::COL_URL_PATH]) {
                $rowData[self::COL_URL_PATH] = $rowData[self::COL_URL];
            }
        }

        if (isset($rowData[self::COL_URL_PATH]) && isset($rowData[self::COL_URL])) {
            $rowData[self::COL_URL] = trim($rowData[self::COL_URL]);
            $rowData[self::COL_URL_PATH] = trim($rowData[self::COL_URL_PATH]);
```

```
        }

        if (isset($rowData[self::COL_URL_PATH]) &&
isset($rowData[self::COL_URL])
            && empty($rowData[self::COL_URL_PATH]) &&
empty($rowData[self::COL_URL])
        ) {
            $rowData = $this->getUrlKeyFromName($rowData);
            if ((isset($rowData[self::COL_URL]) &&
!empty($rowData[self::COL_URL])
                && in_array($rowData[self::COL_URL], $this-
>urlComparableList['url_key'][$storeCode]))
                || ($this-
>isUrlRequestPathDuplicateMage($rowData[self::COL_URL], $rowData))
            ) {
                $rowData = $this->generateUrlKeyOrError($rowData,
$storeCode, $rowNum);
            } else {
                $this->urlComparableList['url_key'][$storeCode][] =
$rowData[self::COL_URL] ?? '';
            }
        } else {
            if (isset($rowData[self::COL_URL_PATH]) &&
empty($rowData[self::COL_URL_PATH])) {
                if (($this->isRootCategory($rowData) &&
isset($rowData[self::COL_URL])
                    && !empty($rowData[self::COL_URL])
                    && in_array($rowData[self::COL_URL], $this-
>urlComparableList['url_key'][$storeCode]))
                    || ($this-
>isUrlRequestPathDuplicateMage($rowData[self::COL_URL_PATH],
$rowData))
                ) {
                    $rowData = $this->generateUrlKeyOrError($rowData,
$storeCode, $rowNum);
                } else {
                    $this->urlComparableList['url_key'][$storeCode][]
= $rowData[self::COL_URL] ?? '';
                }
            } else {
                if (isset($rowData[self::COL_URL_PATH])) {
                    if ((isset($rowData[self::COL_URL_PATH]) &&
!empty($rowData[self::COL_URL_PATH])
                        && in_array($rowData[self::COL_URL_PATH],
$this->urlComparableList['url_path'][$storeCode]))
                        || ($this-
>isUrlRequestPathDuplicateMage($rowData[self::COL_URL_PATH],
$rowData))
                    ) {
                        $rowData = $this-
>generateUrlKeyOrError($rowData, $storeCode, $rowNum);
                    } else {
                        $this-
>urlComparableList['url_path'][$storeCode][] =
$rowData[self::COL_URL_PATH] ?? '';
                    }
                }
            }
        }

        return $rowData;
    }

    /**
     * @param $rowData
     * @param $storeCode
     * @param $rowNum
     * @return mixed
     */
    protected function generateUrlKeyOrError($rowData, $storeCode,
$rowNum)
    {
        $this->foundDuplicate[$storeCode][] =
$rowData[self::COL_NAME];
        if ($this->_parameters['generate_url'] == 1) {
            $rowData = $this->generateUrlKeyProcess($rowData,
$storeCode);
        } else {
            $message = 'category with name: %1 not imported because
its url is not unique.';
            $this->addLogWriteln(__($message,
$rowData[self::COL_NAME]), $this->output, 'error');
            $this->addRowError(__($message,
$rowData[self::COL_NAME]), $rowNum);
        }
        return $rowData;
    }

    /**
     * @param $rowData
     * @return mixed
     */
    protected function getUrlKeyFromName($rowData)
    {
        if (isset($rowData[self::COL_NAME]) &&
!empty($rowData[self::COL_NAME])) {
            $separator = $this-
>_parameters['category_levels_separator'];
            $nameCategories = explode($separator,
$rowData[self::COL_NAME]);
            $nameCategory = $nameCategories[count($nameCategories)-
1];
            $nameCategory = preg_replace('/\s+/', '-',
trim($nameCategory));
            $urlKey = mb_strtolower($nameCategory, 'UTF-8');
            $rowData[self::COL_URL] = $urlKey;
        }
        return $rowData;
    }

    /**
     * @param $rowData
     * @return mixed
     */
    protected function urlPathSlashTrim($rowData)
    {
        if (isset($rowData[self::COL_URL_PATH]) &&
!empty($rowData[self::COL_URL_PATH])) {
            if (substr($rowData[self::COL_URL_PATH], -1) == '/') {
                $rowData[self::COL_URL_PATH] =
substr($rowData[self::COL_URL_PATH], 0, -1);
            }
        }
        return $rowData;
    }

    /**
     * @param $rowData
     * @return bool
     */
    protected function isRootCategory($rowData)
    {
        $result = false;
        if (isset($rowData[self::COL_PATH]) &&
!empty($rowData[self::COL_PATH])) {
            $countSlash = substr_count($rowData[self::COL_PATH],
'/');
            if ($countSlash == 1) {
                $result = true;
            }
        } else {
            $name = isset($rowData[self::COL_NAME]) ?
$rowData[self::COL_NAME] : '';
            if ($name == 'Root Catalog') {
                $result = false;
            } else {
                $countSlash = substr_count($name, '/');
                if ($countSlash == 0) {
                    $result = true;
                }
            }
        }
        return $result;
    }

    /**
     * @param $storeCode
     * @param $rowData
     * @param $rowNum
     * @return mixed
     */
    protected function checkCategoryIfParentIsDuplicate($storeCode,
$rowData, $rowNum)
    {
        if (isset($this->foundDuplicate[$storeCode])) {
            foreach ($this->foundDuplicate[$storeCode] as $key =>
$categoryDuplicate) {
                $position = strpos($rowData[self::COL_NAME],
$categoryDuplicate);
                if ($position == 0) {
                    if ($this->_parameters['generate_url'] == 1) {
                        $rowData['is_url_path_generated'] = 1;
                    } else {
                        $message = 'category with name: %1 not
imported because its url is not unique.';
                        $this->addLogWriteln(__($message,
$rowData[self::COL_NAME]), $this->output, 'error');
                        $this->addRowError(__($message,
$rowData[self::COL_NAME]), $rowNum);
                    }
                    break;
                }
            }
        }
        return $rowData;
    }

    /**
     * @param $rowData
     * @param $storeCode
     * @return mixed
     */
    private function generateUrlKeyProcess($rowData, $storeCode)
    {
        if ($rowData[self::COL_NAME] != 'Root Catalog') {
            if (!empty($rowData[self::COL_URL_PATH])) {
                $oldUrl = $rowData[self::COL_URL_PATH];
            } else {
                $oldUrl = $rowData[self::COL_URL];
            }

            $i = 1;
            $newUrl = $this->generateUrl($oldUrl, $i);
            $generation = true;
            while ($generation) {
                if ($this->isUrlRequestPathDuplicateMage($newUrl,
$rowData)) {
                    $i++;
                    $newUrl = $this->generateUrl($oldUrl, $i);
                } else {
```

```
                    if ($this->isUrlDuplicateComparableList($newUrl,
$storeCode)) {
                        $i++;
                        $newUrl = $this->generateUrl($oldUrl, $i);
                    } else {
                        $this->addKeyInComparableList($newUrl,
$storeCode, $rowData);
                        $rowData['is_url_path_generated'] = 1;
                        $generation = false;
                    }
                }
            }

            $rowData[self::COL_URL_PATH] = $newUrl;
            $newUrl=explode('/', $newUrl);
            $newUrlKey = $newUrl[count($newUrl)-1];
            $rowData[self::COL_URL] = $this-
>formatUrlKey($newUrlKey);
        }

        return $rowData;
    }

    /**
     * @param $url
     * @param $i
     * @return string
     */
    protected function generateUrl($url, $i)
    {
        return $url.$i;
    }

    /**
     * @param $url
     * @param $rowData
     * @return bool
     */
    protected function isUrlRequestPathDuplicateMage($url, $rowData)
    {
        $result = false;
        $storeId = $this->getStoreIdByCode($rowData);
        if (!empty($this->urlRequestPathStoreId)) {
            if (isset($this->urlRequestPathStoreId[$url][$storeId])
                || isset($this-
>urlRequestPathStoreId[$url.'.html'][$storeId])
            ) {
                $result = true;
            }
        }
        return $result;
    }

    /**
     * @param $newUrl
     * @param $storeCode
     * @return bool
     */
    protected function isUrlDuplicateComparableList($newUrl,
$storeCode)
    {
        $result = false;
        if (!empty($this->urlComparableList)) {
            if (in_array($newUrl, $this-
>urlComparableList['url_path'][$storeCode])
                || isset($this-
>urlComparableList['new_generated']['url_path'][$storeCode][$newUrl])
            ) {
                $result = true;
            } else {
                if (in_array($newUrl, $this-
>urlComparableList['url_key'][$storeCode])
                    || isset($this-
>urlComparableList['new_generated']['url_key'][$storeCode][$newUrl])
                ) {
                    $result = true;
                }
            }
        }
        return $result;
    }

    /**
     * @param $rowData
     * @return int
     */
    protected function getStoreIdByCode($rowData)
    {
        $storeId = 1;
        if (!empty($rowData[self::COL_STORE])) {
            if (isset($this->nameToId[$rowData[self::COL_STORE]])) {
                $storeId = $this-
>nameToId[$rowData[self::COL_STORE]];
            }
        }
        return $storeId;
    }

    /**
     * @return array
     */
    protected function initRequestPathStoreId()
    {
        $resource = $this->getResource();
        $select = $this->_connection->select()->from(
            ['url_rewrite' => $resource->getTable('url_rewrite')],
            ['request_path', 'store_id']
        );
        $requestPaths = $this->_connection->fetchAll(
            $select
        );
        if (!empty($requestPaths)) {
            foreach ($requestPaths as $key => $data) {
                $this-
>urlRequestPathStoreId[$data['request_path']][$data['store_id']] = 1;
            }
        }
        return $this->urlRequestPathStoreId;
    }

    /**
     * @param $newUrl
     * @param $storeCode
     * @param $rowData
     */
    protected function addKeyInComparableList($newUrl, $storeCode,
$rowData)
    {
        if (!empty($rowData[self::COL_URL_PATH])) {
            $this-
>urlComparableList['new_generated']['url_path'][$storeCode][$newUrl] =
1;
        } else {
            if (!empty($rowData[self::COL_URL])) {
                $this-
>urlComparableList['new_generated']['url_key'][$storeCode][$newUrl] =
1;
            }
        }
    }

    /**
     * @param $url
     * @param $rowData
     * @return bool
     */
    public function checkUrlKeyDuplicates($url, $rowData)
    {
        $result = false;
        $storeId = $this->getStoreIdByCode($rowData);

        $resource = $this->getResource();
        $select = $this->_connection->select()->from(
            ['url_rewrite' => $resource->getTable('url_rewrite')],
            ['request_path', 'store_id']
        )->joinLeft(
            ['cpe' => $resource->getTable('catalog_product_entity')],
            'cpe.entity_id = url_rewrite.entity_id'
        )->where('request_path LIKE "%' . $url . '"')
            ->orWhere('request_path LIKE "%' . $url . '.html"')
            ->where('store_id IN (?)', $storeId);
        $urlKeyDuplicates = $this->_connection->fetchAssoc(
            $select
        );
        if (!empty($urlKeyDuplicates)) {
            $result = true;
        }
        return $result;
    }

    protected function parseAdditionalAttributes($attributes)
    {
        return empty($this->_parameters[Import::FIELDS_ENCLOSURE])
            ? $this->parseAttributesWithoutWrappedValues($attributes)
            : $this->parseAttributesWithWrappedValues($attributes);
    }

    private function parseAttributesWithoutWrappedValues($data)
    {
        $attributeNameValuePairs = explode(
            $this->getMultipleValueSeparator(),
            $data
        );
        $result = [];
        $code = '';
        foreach ($attributeNameValuePairs as $attributeData) {
            //process case when attribute has
ImportModel::DEFAULT_GLOBAL_MULTI_VALUE_SEPARATOR inside its value
            if (strpos($attributeData,
self::PAIR_NAME_VALUE_SEPARATOR) === false) {
                if (!$code) {
                    continue;
                }
                $result[$code] .= $this->getMultipleValueSeparator()
. $attributeData;
                continue;
            }
            list($code, $value) = explode(
                self::PAIR_NAME_VALUE_SEPARATOR,
                $attributeData,
                2
            );
            $code = mb_strtolower($code);
            $result[$code] = $value;
        }
        return $result;
    }

    private function parseAttributesWithWrappedValues($data)
    {
```

```
        $attributesArray = [];
        preg_match_all(
            '~((?:[a-zA-Z0-9_])+)="((?:[^"]|""|"'
            . $this->getMultiLineSeparatorForRegexp()
            . '")+)"+~',
            $data,
            $matches
        );
        foreach ($matches[1] as $i => $attributeCode) {
            $attribute = $this
                ->retrieveAttributeByCode($attributeCode);
            $value = 'multiselect' != $attribute->getFrontendInput()
                ? str_replace('""', '"', $matches[2][$i])
                : '"' . $matches[2][$i] . '"';
            $attributesArray[mb_strtolower($attributeCode)] = $value;
        }
        return $attributesArray;
    }

    public function getMultipleValueSeparator()
    {
        if (!empty($this-
>_parameters[Import::FIELD_FIELD_MULTIPLE_VALUE_SEPARATOR])) {
            return $this-
>_parameters[Import::FIELD_FIELD_MULTIPLE_VALUE_SEPARATOR];
        }
        return Import::DEFAULT_GLOBAL_MULTI_VALUE_SEPARATOR;
    }

    private function getMultiLineSeparatorForRegexp()
    {
        if (!$this->multiLineSeparatorForRegexp) {
            $this->multiLineSeparatorForRegexp =
in_array(self::PSEUDO_MULTI_LINE_SEPARATOR,
str_split('[\^$.|?*+(){}'))
                ? '\\' . self::PSEUDO_MULTI_LINE_SEPARATOR
                : self::PSEUDO_MULTI_LINE_SEPARATOR;
        }
        return $this->multiLineSeparatorForRegexp;
    }

    /**
     * @param $rowData
     * @param $rowNum
     * @return mixed
     */
    public function customFieldsMapping($rowData, $rowNum)
    {
        if (isset($rowData[self::COL_NAME])) {
            $actualName = \explode(self::DELIMITER_CATEGORY,
$rowData[self::COL_NAME]);
            $rowData['_actual_name'] = \end($actualName);
        }

        if (Import::BEHAVIOR_APPEND == $this->getBehavior()
            || Import::BEHAVIOR_REPLACE == $this->getBehavior()
        ) {
            $rowData = $this->findUrlKeyDuplicates($rowData,
$rowNum);
        }

        return $rowData;
    }

    public function retrieveAttributeByCode($attrCode)
    {
        /** @var string $attrCode */
        $attrCode = mb_strtolower($attrCode);

        if (!isset($this->attributeCache[$attrCode])) {
            $this->attributeCache[$attrCode] = $this->getResource()-
>getAttribute($attrCode);
        }

        return $this->attributeCache[$attrCode];
    }

    public function changeData($rowData)
    {
        foreach ($this->customAttr as $value) {
            if (!array_key_exists($value, $rowData)) {
                continue;
            }
            $rowData[$value] = stripslashes($rowData[$value]);
        }

        foreach ($rowData as $key => $value) {
            if (isset($this->attrData[$key])) {
                $h = $this->attrData[$key];
                if ($h['frontend_input'] == 'select') {
                    $data = $this->retrieveAttributeByCode($key);
                    if (!empty($data->getOptions())) {
                        foreach ($data->getOptions() as
$valueOptions) {
                            $valueData = $valueOptions->getData();
                            if ($valueData['label'] == $value) {
                                $rowData[$key] = $valueData['value'];
                            }
                        }
                    }
                }
            }
        }

        return $rowData;
    }

    protected function getResource()
    {
        if (!$this->resource) {
            $this->resource = $this->resourceFactory->create();
        }
        return $this->resource;
    }

    protected function prepareImagesFromSource($bunch)
    {
        $image = 'image';

        foreach ($bunch as $rowNum => &$rowData) {
            if (empty($rowData[$image])) {
                continue;
            }
            $importImage = $rowData[$image];
            $imageSting = mb_strtolower(preg_replace('/[^a-z0-9\._-
]+/i', '', $importImage));
            if ($this->sourceType) {
                $this->sourceType->importImageCategory($importImage,
$imageSting);
            }
            $imageArr = $imageSting;

            $rowData[$image] = $imageArr;
        }

        return $bunch;
    }

    protected function _initSourceType($type)
    {
        if (!$this->sourceType) {
            $this->sourceType = $this->additional-
>getSourceModelByType($type);
            $this->sourceType->setData($this->_parameters);
        }
    }

    /**
     * Returns initial Categories set in initCategories() method
     *
     * @return array
     */
    public function getInitialCategories()
    {
        return $this->categories;
    }

    /**
     * Format URL key from name or defined key
     *
     * @param string $str
     * @return string
     */
    public function formatUrlKey($str)
    {
        return $this->filterManager->translitUrl($str);
    }

    /**
     * @param $categoryId
     * @return mixed|null
     */
    public function getCategoryById($categoryId)
    {
        return $this->categoriesCache[$categoryId] ?? null;
    }
}
```

```
<?php

namespace Ivan\ImportExport\Model\Import;

use Ivan\ImportExport\Helper\Additional;
use Ivan\ImportExport\Traits\Import\Entity as ImportTrait;
use Magento\Cms\Api\BlockRepositoryInterface;
use Magento\Cms\Api\Data\BlockInterface;
use Magento\Cms\Model\BlockFactory as CmsBlockFactory;
use Magento\Cms\Model\ResourceModel\Block\CollectionFactory;
use Magento\Cms\Model\ResourceModel\BlockFactory;
use Magento\Eav\Model\Config;
use Magento\Framework\App\ResourceConnection;
use Magento\Framework\Event\ManagerInterface;
use Magento\Framework\Json\Helper\Data;
use Magento\Framework\Registry;
use Magento\Framework\Stdlib\StringUtils;
use Magento\ImportExport\Model\Import;
use Magento\ImportExport\Model\Import\Entity\AbstractEntity;
use
Magento\ImportExport\Model\Import\ErrorProcessing\ProcessingError;
use
Magento\ImportExport\Model\Import\ErrorProcessing\ProcessingErrorAggre
gatorInterface;
use Magento\ImportExport\Model\ResourceModel\Helper;
use Magento\Store\Model\Store;
use Magento\Store\Model\StoreManagerInterface;
use Magento\Theme\Model\ResourceModel\Theme\Collection;
use Symfony\Component\Console\Output\ConsoleOutput;

/**
```

```
 * Class CmsBlock
 *
 * @package Ivan\ImportExport\Model\Import
 */
class CmsBlock extends AbstractEntity
{
    use ImportTrait;

    /**
     * Column category url key.
     */
    const COL_URL = 'identifier';

    /**
     * Column cms block_id.
     */
    const COL_BLOCKID = 'block_id';

    /**
     * Column cms store_view_code.
     */
    const COL_STORE_VIEW_CODE = 'store_view_code';

    /**
     * Core event manager proxy
     *
     * @var ManagerInterface
     */
    protected $eventManager = null;

    /**
     * Flag for replace operation.
     *
     * @var null
     */
    protected $replaceFlag = null;
    protected $_validatedRows = [];
    protected $_processedEntitiesCount = null;

    protected $collectionFactory;
    protected $blockFactory;
    protected $blockRepositoryInterface;

    protected $storeManager;

    protected $resource;
    protected $blocksUrl;

    /**
     * @var Registry
     */
    protected $registry;

    /**
     * Block title to ID hash.
     *
     * @var array
     */
    protected $blocks = [];

    protected $sourceType;
    protected $blockResourceFactory;
    protected $additional;

    protected $blockFields = [
        BlockInterface::BLOCK_ID,
        BlockInterface::CONTENT,
        BlockInterface::CREATION_TIME,
        BlockInterface::IDENTIFIER,
        BlockInterface::IS_ACTIVE,
        BlockInterface::TITLE,
        BlockInterface::UPDATE_TIME
    ];
    /**
     * @var Collection
     */
    protected $themeCollection;

    /**
     * @var Store
     */
    protected $store;

    /**
     * CmsBlock constructor.
     * @param Data $jsonHelper
     * @param \Magento\ImportExport\Helper\Data $importExportData
     * @param \Magento\ImportExport\Model\ResourceModel\Import\Data
$importData
     * @param Config $config
     * @param ResourceConnection $resource
     * @param Helper $resourceHelper
     * @param StringUtils $string
     * @param ProcessingErrorAggregatorInterface $errorAggregator
     * @param CollectionFactory $collectionFactory
     * @param CmsBlockFactory $blockFactory
     * @param ManagerInterface $eventManager
     * @param StoreManagerInterface $storeManager
     * @param BlockRepositoryInterface $blockRepositoryInterface
     * @param ConsoleOutput $output
     * @param Registry $registry
     * @param \Ivan\ImportExport\Model\ResourceModel\Import\Data
$importFireData
     * @param BlockFactory $blockResourceFactory
     * @param Additional $additional
     * @param Collection $themeCollection
     * @param Store $store
     * @throws \Magento\Framework\Exception\LocalizedException
     */
    public function __construct(
        Data $jsonHelper,
        \Magento\ImportExport\Helper\Data $importExportData,
        \Magento\ImportExport\Model\ResourceModel\Import\Data
$importData,
        Config $config,
        ResourceConnection $resource,
        Helper $resourceHelper,
        StringUtils $string,
        ProcessingErrorAggregatorInterface $errorAggregator,
        CollectionFactory $collectionFactory,
        CmsBlockFactory $blockFactory,
        ManagerInterface $eventManager,
        StoreManagerInterface $storeManager,
        BlockRepositoryInterface $blockRepositoryInterface,
        ConsoleOutput $output,
        Registry $registry,
        \Ivan\ImportExport\Model\ResourceModel\Import\Data
$importFireData,
        \Magento\Cms\Model\ResourceModel\BlockFactory
$blockResourceFactory,
        Additional $additional,
        Collection $themeCollection,
        Store $store
    ) {
        $this->collectionFactory = $collectionFactory;
        $this->blockFactory = $blockFactory;
        $this->blockRepositoryInterface = $blockRepositoryInterface;
        $this->storeManager = $storeManager;
        $this->eventManager = $eventManager;
        $this->registry = $registry;
        $this->output = $output;
        $this->blockResourceFactory = $blockResourceFactory;
        $this->additional = $additional;
        parent::__construct(
            $jsonHelper,
            $importExportData,
            $importData,
            $config,
            $resource,
            $resourceHelper,
            $string,
            $errorAggregator
        );
        $this->_dataSourceModel = $importFireData;
        $this->blocksUrl = [];
        $this->themeCollection = $themeCollection;
        $this->store = $store;
        $this->initCMSBlocks();
    }

    /**
     *
     */
    protected function initCMSBlocks()
    {
        if (empty($this->blocks)) {
            $stores = $this->storeManager->getStores();
            $searchStores =
[\Magento\Store\Model\Store::DEFAULT_STORE_ID];
            foreach ($stores as $store) {
                $searchStores[] = $store->getId();
            }
            foreach ($searchStores as $store) {
                $collection = $this->collectionFactory->create();
                $collection->addStoreFilter($store);

                /** @var \Magento\Cms\Model\Block $block */
                foreach ($collection as $block) {
                    $this->blocks[$block->getIdentifier()] = $block-
>getId();
                }
            }
        }
    }

    public function getAllFields()
    {
        return array_unique(
            $this->blockFields
        );
    }

    /**
     * Import data rows.
     *
     * @return boolean
     */
    protected function _importData()
    {
        $this->_validatedRows = null;
        if (Import::BEHAVIOR_DELETE == $this->getBehavior()) {
            $this->deletePages();
        } else {
            /**
             * If user select replace behavior all categories will be
deleted first,
             * then new categories will be saved
             */
            $this->savePagesData();
        }
```

```
        $this->eventManager-
>dispatch('cms_blocks_import_finish_before', ['adapter' => $this]);

        return true;
    }

    /**
     * Delete pages is delete behaviour is selected
     *
     * @return $this
     */
    protected function deletePages()
    {
        $blockId = null;
        while ($bunch = $this->_dataSourceModel->getNextBunch()) {
            foreach ($bunch as $rowNum => $rowData) {
                if (!$this->validateRow($rowData, $rowNum)) {
                    continue;
                }
                if ($this->getErrorAggregator()->hasToBeTerminated())
{
                    $this->getErrorAggregator()-
>addRowToSkip($rowNum);
                    continue;
                }
                if (isset($rowData[self::COL_URL]) && isset($this-
>blocks[$rowData[self::COL_URL]])) {
                    $blockId = (int)$this-
>blocks[$rowData[self::COL_URL]];
                } elseif (isset($rowData[self::COL_BLOCKID])) {
                    $blockId = (int)$rowData[self::COL_BLOCKID];
                }

                if ($blockId) {
                    if ($this->blockFactory->create()->
                    getCollection()-
>addFieldToFilter(self::COL_BLOCKID, $blockId)
                        ->getSize()) {
                        try {
                            $block = $this->blockRepositoryInterface-
>getById($blockId);
                            $this->blockRepositoryInterface-
>delete($block);
                        } catch (\Exception $e) {
                            $this->addRowError(
                                $e->getMessage(),
                                $rowNum
                            );
                        }
                    }
                } else {
                    $this->addRowError(
                        'Cannot delete pages ',
                        $rowNum
                    );
                }
            }
        }
        return $this;
    }

    /**
     * Validate data row.
     *
     * @param array $rowData
     * @param int $rowNum
     * @return boolean
     * @SuppressWarnings(PHPMD.CyclomaticComplexity)
     * @SuppressWarnings(PHPMD.NPathComplexity)
     * @SuppressWarnings(PHPMD.ExcessiveMethodLength)
     */
    public function validateRow(array $rowData, $rowNum)
    {

        if (isset($this->_validatedRows[$rowNum])) {
            // check that row is already validated
            return !$this->getErrorAggregator()-
>isRowInvalid($rowNum);
        }

        $this->_validatedRows[$rowNum] = true;
        $this->_processedEntitiesCount++;

        return !$this->getErrorAggregator()->isRowInvalid($rowNum);
    }

    /**
     * Gather and save information of cms pages entities
     *
     * @return CmsBlock
     * @SuppressWarnings(PHPMD.CyclomaticComplexity)
     * @SuppressWarnings(PHPMD.NPathComplexity)
     * @SuppressWarnings(PHPMD.ExcessiveMethodLength)
     * @SuppressWarnings(PHPMD.UnusedLocalVariable)
     */
    protected function savePagesData()
    {
        if (Import::BEHAVIOR_REPLACE == $this->getBehavior()) {
            $this->deletePages();
        }

        while ($bunch = $this->_dataSourceModel->getNextBunch()) {
            foreach ($bunch as $rowNum => $rowData) {
                $rowData = $this->joinIdenticalyData($rowData);
                $rowData = $this->customChangeData($rowData);

                if (!$this->validateRow($rowData, $rowNum)) {
                    $this->addLogWriteln(
                        __('block with name: %1 is not valided',
$rowData['title']),
                        $this->output,
                        'info'
                    );
                    continue;
                }
                $time = explode(" ", microtime());
                $startTime = $time[0] + $time[1];
                $name = $rowData['title'];

                if (!$this->validateRow($rowData, $rowNum)) {
                    continue;
                }

                if ($this->getErrorAggregator()->hasToBeTerminated())
{
                    $this->getErrorAggregator()-
>addRowToSkip($rowNum);
                    continue;
                }

                if (Import::BEHAVIOR_REPLACE == $this->getBehavior())
{
                    if (isset($rowData['block_id'])) {
                        unset($rowData['block_id']);
                    }
                }

                $rowData = $this->checkUrl($rowData);
                $blockStores = $this->getStore($rowData);

                if ($rowData) {
                    try {
                        $block = $this->blockFactory->create();
                        $block->setData($rowData);
                        $block->setStoreId($blockStores);
                        $block->save();
                    } catch (\Exception $e) {
                        $this->getErrorAggregator()->addError(
                            $e->getCode(),
                            ProcessingError::ERROR_LEVEL_NOT_CRITICAL
,
                            $this->_processedRowsCount,
                            null,
                            $e->getMessage()
                        );
                        $this->_processedRowsCount++;
                    }
                }
                $time = explode(" ", microtime());
                $endTime = $time[0] + $time[1];
                $totalTime = $endTime - $startTime;
                $totalTime = round($totalTime, 5);
                $this->addLogWriteln(__('block with name: %1 ....
%2s', $name, $totalTime), $this->output, 'info');
            }
            $this->eventManager->dispatch(
                'cms_blocks_bunch_save_after',
                ['adapter' => $this, 'bunch' => $bunch]
            );
        }
        return $this;
    }

    /**
     * @param $rowData
     * @return mixed
     */
    protected function checkUrl($rowData)
    {
        if (isset($rowData[self::COL_URL])) {
            $url = $this->searchUrl($rowData[self::COL_URL]);
            $rowData[self::COL_URL] = $url;
            $this->blocksUrl[] = $url;
        }

        return $rowData;
    }

    /**
     * @param $url
     * @return mixed
     */
    protected function searchUrl($url)
    {
        if (in_array($url, $this->blocksUrl)) {
            preg_match_all("/\d+$/i", $url, $out);
            if (isset($out[0][0])) {
                $counter = (int)$out[0][0];
                $url = $this->searchUrl(str_replace($counter,
++$counter, $url));
            } else {
                $url = $url;
            }
        }

        if ($this->checkUrlKeyDuplicates($url)) {
            preg_match_all("/\d+$/i", $url, $out);
            if (isset($out[0][0])) {
                $counter = (int)$out[0][0];
                $url = $this->searchUrl(str_replace($counter,
++$counter, $url));
```

```
            }
        }
        return $url;
    }

    protected function checkUrlKeyDuplicates($urlKeys)
    {
        $resource = $this->getResource();
        $select = $this->_connection->select()->from(
            ['url_rewrite' => $resource->getTable('url_rewrite')],
            ['request_path', 'store_id']
        )->joinLeft(
            ['cpe' => $resource->getTable('cms_block')],
            "cpe.block_id = url_rewrite.entity_id"
        )->where('request_path LIKE "%' . $urlKeys . '%"');
        $urlKeyDuplicates = $this->_connection->fetchAssoc(
            $select
        );

        return count($urlKeyDuplicates);
    }

    protected function getResource()
    {
        if (!$this->resource) {
            $this->resource = $this->blockResourceFactory->create();
        }
        return $this->resource;
    }

    protected function getStore($rowData)
    {
        $storeIds = [];
        if (isset($rowData[self::COL_STORE_VIEW_CODE])) {
            if ($rowData[self::COL_STORE_VIEW_CODE] != '') {
                foreach (explode(',',
$rowData[self::COL_STORE_VIEW_CODE]) as $_storeCode) {
                    if ($_storeCode === 'All') {
                        array_push($storeIds, 0);
                    } else {
                        array_push($storeIds, $this->store-
>load($_storeCode)->getId());
                    }
                }
            } else {
                array_push($storeIds, 0);
            }
        }
        return $storeIds;
    }

    /**
     * @return $this
     * @throws \Magento\Framework\Exception\LocalizedException
     */
    protected function _saveValidatedBunches()
    {
        $source = $this->_getSource();
        $currentDataSize = 0;
        $bunchRows = [];
        $startNewBunch = false;
        $nextRowBackup = [];
        $maxDataSize = $this->_resourceHelper->getMaxDataSize();
        $bunchSize = $this->_importExportData->getBunchSize();

        $source->rewind();
        $this->_dataSourceModel->cleanBunches();
        $file = null;
        $jobId = null;
        if (isset($this->_parameters['file'])) {
            $file = $this->_parameters['file'];
        }
        if (isset($this->_parameters['job_id'])) {
            $jobId = $this->_parameters['job_id'];
        }

        while ($source->valid() || $bunchRows) {
            if ($startNewBunch || !$source->valid()) {
                $this->_dataSourceModel->saveBunches(
                    $this->getEntityTypeCode(),
                    $this->getBehavior(),
                    $jobId,
                    $file,
                    $bunchRows
                );
                $bunchRows = $nextRowBackup;
                $currentDataSize =
strlen(\Zend\Serializer\Serializer::serialize($bunchRows));
                $startNewBunch = false;
                $nextRowBackup = [];
            }

            if ($source->valid()) {
                try {
                    $rowData = $source->current();
                } catch (\InvalidArgumentException $e) {
                    $this->addRowError($e->getMessage(), $this-
>_processedRowsCount);
                    $this->_processedRowsCount++;
                    $source->next();
                    continue;
                }

                $this->_processedRowsCount++;
                $rowData = $this->customBunchesData($rowData);
                $rowSize = strlen($this->jsonHelper-
>jsonEncode($rowData));

                $isBunchSizeExceeded = $bunchSize > 0 &&
count($bunchRows) >= $bunchSize;

                if ($currentDataSize + $rowSize >= $maxDataSize ||
$isBunchSizeExceeded) {
                    $startNewBunch = true;
                    $nextRowBackup = [$source->key() => $rowData];
                } else {
                    $bunchRows[$source->key()] = $rowData;
                    $currentDataSize += $rowSize;
                }

                $source->next();
            }
        }
        return $this;
    }

    /**
     * EAV entity type code getter.
     *
     * @return string
     */
    public function getEntityTypeCode()
    {
        return 'cms_block';
    }
}
```

```
<?php

namespace Ivan\ImportExport\Model\Import;

use Exception;
use Ivan\ImportExport\Model\IntegrationFactory;
use Ivan\ImportExport\Model\ResourceModel\Import\Data as ImportData;
use Ivan\ImportExport\Traits\Import\Entity as ImportTrait;
use InvalidArgumentException;
use Magento\Cms\Api\PageRepositoryInterface;
use Magento\Cms\Model\ResourceModel\Page as PageResource;
use Magento\Framework\App\Config\ScopeConfigInterface;
use Magento\Framework\App\ResourceConnection;
use Magento\Framework\Exception\LocalizedException;
use Magento\Framework\Exception\NoSuchEntityException;
use Magento\Framework\Serialize\SerializerInterface;
use Magento\ImportExport\Helper\Data as ImportExportData;
use Magento\ImportExport\Model\Import;
use Magento\ImportExport\Model\Import\AbstractEntity;
use
Magento\ImportExport\Model\Import\ErrorProcessing\ProcessingError;
use Magento\ImportExport\Model\ImportFactory;
use Magento\ImportExport\Model\ResourceModel\Helper;
use Magento\Store\Api\StoreRepositoryInterface;
use Magento\VersionsCms\Api\Data\HierarchyNodeInterface as
NodeInterface;
use Magento\VersionsCms\Helper\Hierarchy as HierarchyHelper;
use Magento\VersionsCms\Model\Hierarchy\Node;
use Magento\VersionsCms\Model\Hierarchy\NodeRepository;
use Magento\VersionsCms\Model\ResourceModel\Hierarchy\Node as
NodeResource;
use Magento\VersionsCms\Model\ResourceModel\Hierarchy\Node\Collection
as NodeCollection;
use Zend\Serializer\Serializer;

/**
 * Class ContentHierarchy
 *
 * @package Ivan\ImportExport\Model\Import
 */
class ContentHierarchy extends AbstractEntity implements
ImportAdapterInterface
{
    use ImportTrait;

    /**
     * Entity type code
     */
    const ENTITY_TYPE_CODE = 'content_hierarchy';

    /**
     * List of available behaviors
     *
     * @var string[]
     */
    protected $_availableBehaviors = [
        Import::BEHAVIOR_APPEND,
        Import::BEHAVIOR_REPLACE,
        Import::BEHAVIOR_DELETE,
    ];

    /**
     * Import export data
     *
     * @var ImportExportData
     */
    protected $_importExportData;

    /**
     * @var Helper
     */
    protected $_resourceHelper;
```

```
    /**
     * @var StoreRepositoryInterface
     */
    protected $storeRepository;

    /**
     * @var ResourceConnection
     */
    protected $resource;

    /**
     * @var HierarchyHelper
     */
    public $hierarchyHelper;

    /**
     * @var NodeResource
     */
    private $nodeResource;

    /**
     * @var NodeRepository
     */
    private $nodeRepository;

    /**
     * @var IntegrationFactory
     */
    private $integrationFactory;

    /**
     * @var PageResource
     */
    private $pageResource;

    /**
     * @var PageRepositoryInterface
     */
    private $pageRepository;

    /**
     *
     * @var array
     */
    private $contentHierarchy;

    /**
     * @var array
     */
    private $contentHierarchyOld;

    /**
     * Json Serializer
     *
     * @var SerializerInterface
     */
    protected $serializer;

    /**
     * @var array
     */
    private $nodeScopes = [
        Node::NODE_SCOPE_DEFAULT,
        Node::NODE_SCOPE_STORE,
        Node::NODE_SCOPE_WEBSITE,
    ];

    /**
     * @var array
     */
    private $nodeFields = [
        NodeInterface::NODE_ID,
        NodeInterface::PARENT_NODE_ID,
        NodeInterface::PAGE_ID,
        NodeInterface::IDENTIFIER,
        NodeInterface::LABEL,
        NodeInterface::LEVEL,
        NodeInterface::SORT_ORDER,
        NodeInterface::REQUEST_URL,
        NodeInterface::XPATH,
        NodeInterface::SCOPE,
        NodeInterface::SCOPE_ID,
    ];

    /**
     * @var array
     */
    private $metadataFields = [];

    /**
     * Error codes
     */
    const ERROR_IDENTIFIERS_IS_EMPTY = 'idAndRequestUrlIsEmpty';
    const ERROR_CMS_PAGE_NOT_FOUND = 'cmsPageNotFound';
    const ERROR_PARENT_NOT_FOUND = 'parentNodeNotFound';

    /**
     * Validation failure message template definitions
     *
     * @var array
     */
    protected $_messageTemplates = [
        self::ERROR_IDENTIFIERS_IS_EMPTY => 'Columns %s is empty',
        self::ERROR_CMS_PAGE_NOT_FOUND => 'The CMS page with the
\'%s\' ID doesn\'t exist',
        self::ERROR_PARENT_NOT_FOUND => 'The CMS node with the \'%s\'
ID doesn\'t exist',
    ];

    /**
     * ContentHierarchy constructor.
     *
     * @param Context $context
     * @param ScopeConfigInterface $scopeConfig
     * @param ImportFactory $importFactory
     * @param ImportExportData $importExportData
     * @param ImportData $importData
     * @param PageResource $pageResource
     * @param PageRepositoryInterface $pageRepository
     * @param StoreRepositoryInterface $storeRepository
     * @param IntegrationFactory $integrationFactory
     * @param array $data
     */
    public function __construct(
        Context $context,
        ScopeConfigInterface $scopeConfig,
        ImportFactory $importFactory,
        ImportExportData $importExportData,
        ImportData $importData,
        PageResource $pageResource,
        PageRepositoryInterface $pageRepository,
        StoreRepositoryInterface $storeRepository,
        IntegrationFactory $integrationFactory,
        array $data = []
    ) {
        parent::__construct(
            $context->getStringUtils(),
            $scopeConfig,
            $importFactory,
            $context->getResourceHelper(),
            $context->getResource(),
            $context->getErrorAggregator(),
            $data
        );
        $this->integrationFactory = $integrationFactory;
        $this->hierarchyHelper = $data['hierarchyHelper'];
        $this->nodeResource = $data['nodeResource'];
        $this->nodeRepository = $data['nodeRepository'];
        $this->pageResource = $pageResource;
        $this->pageRepository = $pageRepository;
        $this->storeRepository = $storeRepository;
        $this->_importExportData = $importExportData;
        $this->_resourceHelper = $context->getResourceHelper();
        $this->_dataSourceModel = $importData;
        $this->resource = $context->getResource();
        $this->output = $context->getOutput();
           $this->serializer = $context->getSerializer();
        $this->metadataFields = $this->hierarchyHelper-
>getMetadataFields();

        foreach ($this->_messageTemplates as $errorCode => $message)
{
            $this->getErrorAggregator()-
>addErrorMessageTemplate($errorCode, $message);
        }
        $this->initContentHierarchy();
    }

    /**
     * Initialize Content Hierarchy
     */
    private function initContentHierarchy()
    {
        if (empty($this->contentHierarchy)) {
            /** @var NodeCollection $nodeCollection */
            $nodeCollection = $this->integrationFactory-
>create(NodeCollection::class);
            foreach ($nodeCollection as $node) {
                $requestUrl = $node->getRequestUrl();
                $scope = $node->getScope();
                $scopeId = $node->getScopeId();
                if (!empty($requestUrl)) {
                    $this-
>contentHierarchy[$requestUrl][$scope][$scopeId] = $node->getId();
                } else {
                    $this->contentHierarchy[$scope][$scopeId] =
$node->getId();
                }
            }
        }
    }

    /**
     * Imported entity type code getter.
     *
     * @return string
     */
    public function getEntityTypeCode()
    {
        return self::ENTITY_TYPE_CODE;
    }

    /**
     * Retrieve All Fields Source
     *
     * @return array
     */
    public function getAllFields()
    {
```

```
        return array_unique($this->nodeFields + $this->metadataFields);
    }

    /**
     * Import data rows
     *
     * @return boolean
     * @throws LocalizedException
     */
    protected function _importData()
    {
        $this->_validatedRows = null;
        if (Import::BEHAVIOR_DELETE == $this->getBehavior()) {
            $this->delete();
        } elseif (Import::BEHAVIOR_REPLACE == $this->getBehavior()) {
            $this->replaceProcess();
        } else {
            $this->save();
        }

        return true;
    }

    /**
     * @return $this
     * @throws LocalizedException
     */
    protected function replaceProcess()
    {
        if (empty($this->contentHierarchy)) {
            $this->addLogWriteln(
                __('Node can\'t be replaced. Firstly add before replace'),
                $this->output,
                'error'
            );
        } else {
            $this->delete();
            $this->save();
        }

        return $this;
    }

    /**
     * Delete Content Hierarchy if delete behaviour is selected
     *
     * @return $this
     */
    private function delete()
    {
        $idsToDelete = [];
        while ($bunch = $this->_dataSourceModel->getNextBunch()) {
            foreach ($bunch as $rowNum => $rowData) {
                if ($this->validateRow($rowData, $rowNum)) {
                    $idsToDelete = array_merge($idsToDelete, $this->getProcessedIds($rowData));
                }
                if (!$rowData || $this->validateRow($rowData, $rowNum)) {
                    continue;
                }
                if ($this->getErrorAggregator()->hasToBeTerminated()) {
                    $this->getErrorAggregator()->addRowToSkip($rowNum);
                    continue;
                }
            }

            if ($idsToDelete) {
                try {
                    $this->nodeResource->dropNodes($idsToDelete);
                    $this->countItemsDeleted += count($idsToDelete);
                    if (Import::BEHAVIOR_REPLACE == $this->getBehavior()) {
                        $this->contentHierarchyOld = $this->contentHierarchy;
                    }
                    foreach ($this->contentHierarchy as $requestUrl => $hierarchyByUrl) {
                        foreach ($hierarchyByUrl as $scope => $hierarchyByScope) {
                            if (is_array($hierarchyByScope)) {
                                foreach ($hierarchyByScope as $scopeId => $id) {
                                    if (in_array($id, $idsToDelete)) {
                                        unset($this->contentHierarchy[$requestUrl][$scope][$scopeId]);
                                    }
                                }
                            }
                        }
                    }
                } catch (Exception $e) {
                    $this->addLogWriteln(__('Nodes can\'t be deleted.'), $this->output, 'error');
                }
            }
        }

        if ($this->getDeletedItemsCount()) {
            $this->addLogWriteln(__('Deleted: %1 nodes.', $this->getDeletedItemsCount()), $this->output, 'info');
        }

        return $this;
    }

    /**
     * Gather and save information of Hierarchy nodes
     *
     * @return $this
     * @throws LocalizedException
     */
    public function save()
    {
        while ($bunch = $this->_dataSourceModel->getNextBunch()) {
            foreach ($bunch as $rowNum => $rowData) {
                $this->_processedRowsCount++;
                $rowData = $this->joinIdenticalyData($rowData);
                $rowData = $this->customChangeData($rowData);
                $rowData = $this->prepareRowData($rowData);
                if (!$this->validateRow($rowData, $rowNum)) {
                    continue;
                }

                if ($this->getErrorAggregator()->hasToBeTerminated()) {
                    $this->getErrorAggregator()->addRowToSkip($rowNum);
                    continue;
                }

                if (Import::BEHAVIOR_REPLACE == $this->getBehavior()
                    && !$this->isDataForReplace($rowData)) {
                    continue;
                }

                if ($rowData) {
                    try {
                        $nodeData = $this->prepareNodeData($rowData);
                        $this->createNode($nodeData);
                        $this->countItemsUpdated++;
                    } catch (Exception $e) {
                        $this->getErrorAggregator()->addError(
                            $e->getCode(),
                            ProcessingError::ERROR_LEVEL_NOT_CRITICAL,
                            $this->getProcessedRowsCount(),
                            null,
                            $e->getMessage()
                        );
                    }
                }
            }

            if ($this->getUpdatedItemsCount()) {
                $this->addLogWriteln(__('Imported: %1 Nodes.', $this->getUpdatedItemsCount()), $this->output);
            }
        }

        return $this;
    }

    /**
     * @param array $rowData
     * @return bool
     */
    private function isDataForReplace(array $rowData)
    {
        $requestUrl = $rowData[NodeInterface::REQUEST_URL];
        $scope = $rowData[NodeInterface::SCOPE];
        $scopeId = $rowData[NodeInterface::SCOPE_ID];
        if (isset($this->contentHierarchyOld[$requestUrl][$scope][$scopeId])) {
            $result = true;
        } else {
            $result = false;
        }

        return $result;
    }

    /**
     * Retrieve Node id's for delete
     *
     * @param array $rowData
     * @return array
     */
    private function getProcessedIds(array $rowData)
    {
        $processedIds = [];
        $requestUrl = $rowData[NodeInterface::REQUEST_URL] ?? null;
        $scope = $rowData[NodeInterface::SCOPE] ?? null;
        $scopeId = $rowData[NodeInterface::SCOPE_ID] ?? null;
        $nodeId = $rowData[NodeInterface::NODE_ID] ?? null;

        if ($requestUrl && $scope && $scopeId && isset($this->contentHierarchy[$requestUrl][$scope][$scopeId])) {
            $processedIds[] = $this->contentHierarchy[$requestUrl][$scope][$scopeId];
        } elseif ($requestUrl && $scope && isset($this->contentHierarchy[$requestUrl][$scope])) {
            foreach ($this->contentHierarchy[$requestUrl][$scope] as $id) {
```

```
                $processedIds[] = $id;
            }
        } elseif ($requestUrl && isset($this-
>contentHierarchy[$requestUrl])) {
            foreach ($this->contentHierarchy[$requestUrl] as $data) {
                foreach ($data as $id) {
                    $processedIds[] = $id;
                }
            }
        } elseif ($nodeId) {
            $processedIds[] = $nodeId;
        }

        return $processedIds;
    }

    /**
     * Create Content Hierarchy Node
     *
     * @param $rowData
     * @return NodeInterface
     * @throws LocalizedException
     */
    private function createNode($rowData)
    {
        $node = $this->integrationFactory->create(Node::class)-
>setData($rowData);
        $newNode = $this->nodeRepository->save($node);
        $newNodeId = $newNode->getId();
        $newNodeRequestUrl = $newNode->getRequestUrl();
        $newNodeScope = $newNode->getScope();
        $newNodeScopeId = $newNode->getScopeId();
        $this-
>contentHierarchy[$newNodeRequestUrl][$newNodeScope][$newNodeScopeId]
= $newNodeId;
        return $newNode;
    }

    /**
     * Validate data row
     *
     * @param array $rowData
     * @param int $rowNumber
     * @return bool
     */
    public function validateRow(array $rowData, $rowNumber)
    {
        if (isset($this->_validatedRows[$rowNumber])) {
            /* check that row is already validated */
            return !$this->getErrorAggregator()-
>isRowInvalid($rowNumber);
        }

        $this->_validatedRows[$rowNumber] = true;
        $this->_processedEntitiesCount++;

        /* behavior selector */
        switch ($this->getBehavior()) {
            case Import::BEHAVIOR_DELETE:
                break;
            case Import::BEHAVIOR_REPLACE:
            case Import::BEHAVIOR_APPEND:
            case AbstractEntity::getDefaultBehavior():
                if (empty($rowData[NodeInterface::NODE_ID]) &&
empty($rowData[NodeInterface::REQUEST_URL])) {
                    $colName = NodeInterface::NODE_ID . __(' and ') .
NodeInterface::REQUEST_URL;
                    $this-
>addRowError(self::ERROR_IDENTIFIERS_IS_EMPTY, $rowNumber, $colName);
                }

                if (!empty($rowData[NodeInterface::NODE_ID]) &&
empty($rowData[NodeInterface::REQUEST_URL])) {
                    if
(!empty($rowData[NodeInterface::PARENT_NODE_ID])) {
                        $parentNodeId =
$rowData[NodeInterface::PARENT_NODE_ID];
                        try {
                            $this->nodeRepository-
>getById($parentNodeId);
                        } catch (NoSuchEntityException $e) {
                            $this-
>addRowError(self::ERROR_PARENT_NOT_FOUND, $rowNumber, $parentNodeId);
                        }
                    }
                }

                if (!empty($rowData[NodeInterface::PAGE_ID])) {
                    $pageId = $rowData[NodeInterface::PAGE_ID];
                    try {
                        $this->pageRepository->getById($pageId);
                    } catch (LocalizedException $e) {
                        $this-
>addRowError(self::ERROR_CMS_PAGE_NOT_FOUND, $rowNumber, $pageId);
                    }
                }
                break;
        }
        return !$this->getErrorAggregator()-
>isRowInvalid($rowNumber);
    }

    /**
     * Prepare row data for update/replace behaviour
     *
     * @param array $rowData
     * @return array
     * @throws LocalizedException
     */
    private function prepareRowData(array $rowData)
    {
        if (!empty($rowData[NodeInterface::IDENTIFIER]) &&
empty($rowData[NodeInterface::REQUEST_URL])) {
            $rowData[NodeInterface::REQUEST_URL] =
$rowData[NodeInterface::IDENTIFIER];
        }

        if (empty($rowData[NodeInterface::SCOPE]) ||
!in_array($rowData[NodeInterface::SCOPE], $this->nodeScopes)) {
            $rowData[NodeInterface::SCOPE] =
Node::NODE_SCOPE_DEFAULT;
            $rowData[NodeInterface::SCOPE_ID] =
Node::NODE_SCOPE_DEFAULT_ID;
        } elseif (!isset($rowData[NodeInterface::SCOPE_ID]) ||
!is_numeric($rowData[NodeInterface::SCOPE_ID])) {
            $rowData[NodeInterface::SCOPE_ID] =
Node::NODE_SCOPE_DEFAULT_ID;
        }

        $pageId = null;
        $identifier = null;
        if (!empty($rowData[NodeInterface::REQUEST_URL])) {
            $identifiers = explode("/",
$rowData[NodeInterface::REQUEST_URL]);
            if (empty($rowData[NodeInterface::LEVEL])) {
                $rowData[NodeInterface::LEVEL] = count($identifiers);
            }
            $identifier = array_pop($identifiers);
            $scope = $rowData[NodeInterface::SCOPE];
            $scopeId = $rowData[NodeInterface::SCOPE_ID];
            $storeId = ($scope == Node::NODE_SCOPE_STORE) ? $scopeId
: 0;
            $pageId = $this->pageResource-
>checkIdentifier($identifier, $storeId);
            if ($pageId) {
                $rowData[NodeInterface::PAGE_ID] = $pageId;
            }
        } elseif (!empty($rowData[NodeInterface::PAGE_ID])) {
            $pageId = $rowData[NodeInterface::PAGE_ID];
            $page = $this->pageRepository->getById($pageId);
            if ($page && $page->getId()) {
                $rowData[NodeInterface::REQUEST_URL] = $page-
>getIdentifier();
            }
        }

        if (!empty($rowData[NodeInterface::PAGE_ID])) {
            $rowData[NodeInterface::IDENTIFIER] = null;
            $rowData[NodeInterface::LABEL] = null;
        } else {
            $rowData[NodeInterface::PAGE_ID] = null;
            if (empty($rowData[NodeInterface::IDENTIFIER])) {
                $rowData[NodeInterface::IDENTIFIER] = $identifier;
            }
            $identifier = $rowData[NodeInterface::IDENTIFIER];
            if (empty($rowData[NodeInterface::LABEL]) &&
!empty($identifier)) {
                $rowData[NodeInterface::LABEL] =
ucwords(str_replace(['-', '_'], ' ', $identifier));
            }
        }

        return $rowData;
    }

    /**
     * Prepare node data for save
     *
     * @param $rowData
     * @return array
     * @throws LocalizedException
     */
    private function prepareNodeData($rowData)
    {
        $scope = $rowData[NodeInterface::SCOPE];
        $scopeId = $rowData[NodeInterface::SCOPE_ID];
        $requestUrl = null;
        $xpath = null;
        $parentNodeId = null;
        $nodeData = [];

        if (!empty($rowData[NodeInterface::REQUEST_URL])) {
            $identifiers = explode("/",
$rowData[NodeInterface::REQUEST_URL]);
            foreach ($identifiers as $identifier) {
                $requestUrl = !empty($requestUrl) ? $requestUrl . "/"
. $identifier : $identifier;
                $xpath = !empty($xpath) ? $xpath . "/" : '';
                if (!isset($this-
>contentHierarchy[$requestUrl][$scope][$scopeId])) {
                    $nodeData = [
                        NodeInterface::NODE_ID => null,
                        NodeInterface::PARENT_NODE_ID =>
$parentNodeId,
                        NodeInterface::XPATH => $xpath,
                        NodeInterface::SCOPE => $scope,
                        NodeInterface::SCOPE_ID => $scopeId,
                    ];
                    if ($rowData[NodeInterface::REQUEST_URL] !=
$requestUrl) {
```

```
                        $storeId = ($scope == Node::NODE_SCOPE_STORE)
? $scopeId : 0;
                        $pageId = $this->pageResource-
>checkIdentifier($identifier, $storeId);
                        $label = ucwords(str_replace(['-', '_'], ' ',
$identifier));
                        $nodeData[NodeInterface::PAGE_ID] = $pageId ?
$pageId : null;
                        $nodeData[NodeInterface::IDENTIFIER] =
$pageId ? null : $identifier;
                        $nodeData[NodeInterface::LABEL] = $pageId ?
null : $label;
                        $nodeData[NodeInterface::LEVEL] =
count(explode("/", $requestUrl));
                        $nodeData[NodeInterface::SORT_ORDER] = 0;
                        $nodeData[NodeInterface::REQUEST_URL] =
$requestUrl;
                        $newNode = $this->createNode($nodeData);
                        $parentNodeId = $newNode->getId();
                        $xpath = $newNode->getXpath();
                    } else {
                        $nodeData[NodeInterface::PAGE_ID] =
$rowData[NodeInterface::PAGE_ID];
                        $nodeData[NodeInterface::IDENTIFIER] =
$rowData[NodeInterface::IDENTIFIER];
                        $nodeData[NodeInterface::LABEL] =
$rowData[NodeInterface::LABEL];
                        $nodeData[NodeInterface::LEVEL] =
$rowData[NodeInterface::LEVEL];
                        $nodeData[NodeInterface::SORT_ORDER] =
$rowData[NodeInterface::SORT_ORDER] ?? 0;
                        $nodeData[NodeInterface::REQUEST_URL] =
$rowData[NodeInterface::REQUEST_URL];
                        $rowData = $this->hierarchyHelper-
>copyMetaData($rowData, $nodeData);
                    }
                } else {
                    /** @var Node $existNode */
                    $existNode = $this->integrationFactory
                        ->create(
                            Node::class,
                            ['data' => [NodeInterface::SCOPE =>
$scope, NodeInterface::SCOPE_ID => $scopeId]]
                        )->loadByRequestUrl($requestUrl);
                    if ($rowData[NodeInterface::REQUEST_URL] !=
$requestUrl) {
                        $parentNodeId = $existNode->getId();
                        $xpath = $existNode->getXpath();
                    } else {
                        unset($this-
>contentHierarchy[$requestUrl][$scope][$scopeId]);
                        unset($rowData[NodeInterface::NODE_ID]);
                        $nodeData = $existNode->getData();
                        foreach ($this->nodeFields as $field) {
                            if (isset($rowData[$field])) {
                                $nodeData[$field] = $rowData[$field];
                                if ($field ==
NodeInterface::PARENT_NODE_ID) {
                                    $nodeData[$field] =
$parentNodeId;
                                }
                                if ($field == NodeInterface::XPATH) {
                                    $nodeData[$field] = $xpath;
                                }
                            }
                        }
                        $rowData = $this->hierarchyHelper-
>copyMetaData($rowData, $nodeData);
                    }
                }
            }
        } elseif (!empty($scope) && !empty($scopeId)) {
            if (!isset($this->contentHierarchy[$scope][$scopeId])) {
                unset($rowData[NodeInterface::NODE_ID]);
            }
            $nodeField = $this->getAllFields();
            foreach ($nodeField as $field) {
                if (!empty($rowData[$field])) {
                    $nodeData[$field] = $rowData[$field];
                }
            }
            $rowData = $nodeData;
        } else {
            $rowData = [];
        }

        return $rowData;
    }

    /**
     * Save Validated Bunches
     *
     * @return $this
     * @throws LocalizedException
     */
    protected function _saveValidatedBunches()
    {
        $source = $this->getSource();
        $currentDataSize = 0;
        $bunchRows = [];
        $startNewBunch = false;
        $nextRowBackup = [];
        $maxDataSize = $this->_resourceHelper->getMaxDataSize();
        $bunchSize = $this->_importExportData->getBunchSize();
        $source->rewind();
        $this->_dataSourceModel->cleanBunches();
        $file = null;
        $jobId = null;
        if (isset($this->_parameters['file'])) {
            $file = $this->_parameters['file'];
        }
        if (isset($this->_parameters['job_id'])) {
            $jobId = $this->_parameters['job_id'];
        }

        while ($source->valid() || $bunchRows) {
            if ($startNewBunch || !$source->valid()) {
                $this->_dataSourceModel->saveBunches(
                    $this->getEntityTypeCode(),
                    $this->getBehavior(),
                    $jobId,
                    $file,
                    $bunchRows
                );
                $bunchRows = $nextRowBackup;
                $currentDataSize =
strlen(Serializer::serialize($bunchRows));
                $startNewBunch = false;
                $nextRowBackup = [];
            }

            if ($source->valid()) {
                try {
                    $rowData = $source->current();
                } catch (InvalidArgumentException $e) {
                    $this->addRowError($e->getMessage(), $this-
>getProcessedRowsCount());
                    $this->_processedRowsCount++;
                    $source->next();
                    continue;
                }
                $rowData = $this->customBunchesData($rowData);
                $this->_processedRowsCount++;

                $rowSize = strlen($this->serializer-
>serialize($rowData));

                $isBunchSizeExceeded = $bunchSize > 0 &&
count($bunchRows) >= $bunchSize;

                if ($currentDataSize + $rowSize >= $maxDataSize ||
$isBunchSizeExceeded) {
                    $startNewBunch = true;
                    $nextRowBackup = [$source->key() => $rowData];
                } else {
                    $bunchRows[$source->key()] = $rowData;
                    $currentDataSize += $rowSize;
                }

                $source->next();
            }
        }
        return $this;
    }
}
```

```
<?php

namespace Ivan\ImportExport\Model\Import;

use Ivan\ImportExport\Traits\Import\Entity as ImportTrait;
use Magento\CustomerImportExport\Model\Import\Customer as
MagentoCustomer;
use Symfony\Component\Console\Output\ConsoleOutput;
use \Magento\ImportExport\Model\Import\AbstractEntity;
use Ivan\ImportExport\Model\Import;

/**
 * Class Customer
 *
 * @package Ivan\ImportExport\Model\Import
 */
class Customer extends MagentoCustomer
{
    use ImportTrait;

    protected $_debugMode;

    /**
     * @var array
     */
    protected $superUserList = [];

    /**
     * @var \Magento\Framework\App\ResourceConnection
     */
    protected $_resource;

    /**
     * @param Context $context
     * @param \Magento\Framework\App\Config\ScopeConfigInterface
$scopeConfig
     * @param \Magento\ImportExport\Model\ImportFactory
$importFactory
     * @param ProcessingErrorAggregator $errorAggregator
     * @param \Magento\Store\Model\StoreManagerInterface
$storeManager
     * @param \Magento\ImportExport\Model\Export\Factory
$collectionFactory
```

```
     * @param
\Magento\CustomerImportExport\Model\ResourceModel\Import\Customer\Stor
ageFactory $storageFactory
     * @param
\Magento\Customer\Model\ResourceModel\Attribute\CollectionFactory
$attrCollectionFactory
     * @param \Magento\Customer\Model\CustomerFactory
$customerFactory
     * @param ConsoleOutput $output
     * @param \Ivan\ImportExport\Helper\Data $helper
     * @param array $data
     */
    public function __construct(
        Context $context,
        \Magento\Framework\App\Config\ScopeConfigInterface
$scopeConfig,
        \Magento\ImportExport\Model\ImportFactory $importFactory,
        ProcessingErrorAggregator $errorAggregator,
        \Magento\Store\Model\StoreManagerInterface $storeManager,
        \Magento\ImportExport\Model\Export\Factory
$collectionFactory,
        \Magento\CustomerImportExport\Model\ResourceModel\Import\Cust
omer\StorageFactory $storageFactory,
        \Magento\Customer\Model\ResourceModel\Attribute\CollectionFac
tory $attrCollectionFactory,
        \Magento\Customer\Model\CustomerFactory $customerFactory,
        ConsoleOutput $output,
        \Ivan\ImportExport\Helper\Data $helper,
        array $data = []
    ) {
        parent::__construct(
            $context->getStringUtils(),
            $scopeConfig,
            $importFactory,
            $context->getResourceHelper(),
            $context->getResource(),
            $errorAggregator,
            $storeManager,
            $collectionFactory,
            $context->getConfig(),
            $storageFactory,
            $attrCollectionFactory,
            $customerFactory,
            $data
        );
        $this->_availableBehaviors = [
            \Magento\ImportExport\Model\Import::BEHAVIOR_APPEND,
            \Magento\ImportExport\Model\Import::BEHAVIOR_DELETE,
            \Magento\ImportExport\Model\Import::BEHAVIOR_REPLACE,
        ];
        $this->output = $output;
        $this->_logger = $context->getLogger();
        $this->_debugMode = $helper->getDebugMode();
        $this->_dataSourceModel = $context->getDataSourceModel();
        $this->_resource = $context->getResource();
        $this->_helper = $helper;
    }

    /**
     * @return array
     */
    public function getAllFields()
    {
        $options = array_merge($this->getValidColumnNames(), $this-
>_specialAttributes);
        $options = array_merge($options, $this-
>_permanentAttributes);

        return array_unique($options);
    }

    protected function _importData()
    {
        while ($bunch = $this->_dataSourceModel->getNextBunch()) {
            $entitiesCreate = [];
            $entitiesUpdate = [];
            $entitiesDelete = [];
            $attributesToSave = [];

            foreach ($bunch as $rowNumber => $rowData) {
                $time = explode(" ", microtime());
                $startTime = $time[0] + $time[1];
                $email = $rowData['email'];
                $rowData = $this->joinIdenticalyData($rowData);
                $website = $rowData[self::COLUMN_WEBSITE];
                if (isset($this-
>_newCustomers[strtolower($rowData[self::COLUMN_EMAIL])][$website])) {
                    continue;
                }
                $rowData = $this->customChangeData($rowData);
                if (!$this->validateRow($rowData, $rowNumber)) {
                    $this->addLogWriteln(__('customer with email: %1
is not valided', $email), $this->output, 'info');
                    continue;
                }
                if ($this->getErrorAggregator()->hasToBeTerminated())
{
                    $this->getErrorAggregator()-
>addRowToSkip($rowNumber);
                    continue;
                }

                if
(\Magento\ImportExport\Model\Import::BEHAVIOR_DELETE == $this-
>getBehavior($rowData)) {
                    $entitiesDelete[] = $this->_getCustomerId(
                        $rowData[self::COLUMN_EMAIL],
                        $rowData[self::COLUMN_WEBSITE]
                    );
                }
                if
(\Magento\ImportExport\Model\Import::BEHAVIOR_ADD_UPDATE == $this-
>getBehavior($rowData)
                    ||
\Magento\ImportExport\Model\Import::BEHAVIOR_REPLACE == $this-
>getBehavior($rowData)
                ) {
                    $processedData = $this-
>_prepareDataForUpdate($rowData);
                    if
(\Magento\ImportExport\Model\Import::BEHAVIOR_ADD_UPDATE == $this-
>getBehavior($rowData)) {
                        $entitiesCreate =
array_merge($entitiesCreate,
$processedData[self::ENTITIES_TO_CREATE_KEY]);
                    }
                    $entitiesUpdate = array_merge($entitiesUpdate,
$processedData[self::ENTITIES_TO_UPDATE_KEY]);
                    foreach
($processedData[self::ATTRIBUTES_TO_SAVE_KEY] as $tableName =>
$customerAttributes) {
                        if (!isset($attributesToSave[$tableName])) {
                            $attributesToSave[$tableName] = [];
                        }
                        $attributesToSave[$tableName] =
                            array_diff_key(
                                $attributesToSave[$tableName],
                                $customerAttributes
                            ) + $customerAttributes;
                    }
                }
                $time = explode(" ", microtime());
                $endTime = $time[0] + $time[1];
                $totalTime = $endTime - $startTime;
                $totalTime = round($totalTime, 5);

                $this->addLogWriteln(
                    __('customer with email: %1 .... %2s', $email,
$totalTime),
                    $this->output,
                    'info'
                );
            }
            $this->updateItemsCounterStats($entitiesCreate,
$entitiesUpdate, $entitiesDelete);
            /**
             * Save prepared data
             */
            if ($entitiesCreate || $entitiesUpdate) {
                $this->_saveCustomerEntities($entitiesCreate,
$entitiesUpdate);
            }
            if ($attributesToSave) {
                $this->_saveCustomerAttributes($attributesToSave);
            }
            if ($entitiesDelete) {
                $this->_deleteCustomerEntities($entitiesDelete);
            }
        }

        return true;
    }

    protected function _saveValidatedBunches()
    {
        $source = $this->getSource();
        $bunchRows = [];
        $startNewBunch = false;

        $source->rewind();
        $this->_dataSourceModel->cleanBunches();
        $masterAttributeCode = $this->getMasterAttributeCode();
        $file = null;
        $jobId = null;
        if (isset($this->_parameters['file'])) {
            $file = $this->_parameters['file'];
        }
        if (isset($this->_parameters['job_id'])) {
            $jobId = $this->_parameters['job_id'];
        }
        $isSuperUserList = $this->initSuperUserListProcess();
        while ($source->valid() || count($bunchRows) ||
isset($entityGroup)) {
            if ($startNewBunch || !$source->valid()) {
                /* If the end approached add last validated entity
group to the bunch */
                if (!$source->valid() && isset($entityGroup)) {
                    foreach ($entityGroup as $key => $value) {
                        $bunchRows[$key] = $value;
                    }
                    unset($entityGroup);
                }
                $this->_dataSourceModel->saveBunches(
                    $this->getEntityTypeCode(),
                    $this->getBehavior(),
                    $jobId,
                    $file,
                    $bunchRows
                );
```

```
                $bunchRows = [];
                $startNewBunch = false;
            }
            if ($source->valid()) {
                $valid = true;
                try {
                    $rowData = $source->current();
                    foreach ($rowData as $attrName => $element) {
                        if (!mb_check_encoding($element, 'UTF-8')) {
                            $valid = false;
                            $this->addRowError(
                                AbstractEntity::ERROR_CODE_ILLEGAL_CH
ARACTERS,
                                $this->_processedRowsCount,
                                $attrName
                            );
                        }
                    }
                } catch (\InvalidArgumentException $e) {
                    $valid = false;
                    $this->addRowError($e->getMessage(), $this-
>_processedRowsCount);
                }
                if (!$valid) {
                    $this->_processedRowsCount++;
                    $source->next();
                    continue;
                }
                $rowData = $this->customBunchesData($rowData);
                if ($isSuperUserList) {
                    $this->checkSuperUser($rowData, $source->key());
                }
                if (isset($rowData[$masterAttributeCode]) &&
trim($rowData[$masterAttributeCode])) {
                    /* Add entity group that passed validation to
bunch */
                    if (isset($entityGroup)) {
                        foreach ($entityGroup as $key => $value) {
                            $bunchRows[$key] = $value;
                        }
                        $productDataSize =
strlen(\Zend\Serializer\Serializer::serialize($bunchRows));

                        /* Check if the new bunch should be started
*/
                        $isBunchSizeExceeded = ($this->_bunchSize > 0
&& count($bunchRows) >= $this->_bunchSize);
                        $startNewBunch = $productDataSize >= $this-
>_maxDataSize || $isBunchSizeExceeded;
                    }

                    /* And start a new one */
                    $entityGroup = [];
                }

                if (isset($entityGroup) && $this-
>validateRow($rowData, $source->key())) {
                    /* Add row to entity group */
                    $entityGroup[$source->key()] = $this-
>_prepareRowForDb($rowData);
                } elseif (isset($entityGroup)) {
                    /* In case validation of one line of the group
fails kill the entire group */
                    unset($entityGroup);
                }

                $this->_processedRowsCount++;
                $source->next();
            }
        }
        return $this;
    }

    /**
     * @return bool
     */
    protected function initSuperUserListProcess()
    {
        $result = false;
        if (\Magento\ImportExport\Model\Import::BEHAVIOR_DELETE ==
$this->getBehavior()
            && $this->_connection->isTableExists('company')
        ) {
            $tableName = $this->_resource->getTableName('company');
            $select = $this->_connection->select();
            $select->from(['c' => $tableName], 'c.super_user_id');
            try {
                $data = $this->_connection->fetchAll($select);
            } catch (\Exception $e) {
                $this->_logger->error($e->getMessage());
            }
            if (!empty($data)) {
                foreach ($data as $row) {
                    $this->superUserList[$row['super_user_id']] = 1;
                }
                $result = true;
            }
        }
        return $result;
    }

    /**
     * @param $customerId
     * @return int
     */
    protected function isSuperUser($customerId)
    {
        return isset($this->superUserList[$customerId]) ? 1 : 0;
    }

    /**
     * @param $rowData
     * @param $rowNum
     */
    protected function checkSuperUser($rowData, $rowNum)
    {
        if (isset($rowData[self::COLUMN_EMAIL]) &&
isset($rowData[self::COLUMN_WEBSITE])) {
            $customerId = $this->_getCustomerId(
                $rowData[self::COLUMN_EMAIL],
                $rowData[self::COLUMN_WEBSITE]
            );
            if ($this->isSuperUser($customerId)) {
                $email = $rowData[self::COLUMN_EMAIL];
                $message = 'Cannot delete the company admin. Customer
with email: %1 is company admin.';
                $this->addLogWriteln(__($message, $email), $this-
>output, 'error');
                $this->addRowError(__($message, $email), $rowNum);
            }
        }
    }

    protected function _validateRowForUpdate(array $rowData, $rowNum)
    {
        if ($this->_checkUniqueKey($rowData, $rowNum)) {
            $email = strtolower($rowData[self::COLUMN_EMAIL]);
            $website = $rowData[self::COLUMN_WEBSITE];
            $this->_newCustomers[$email][$website] = false;

            if (!empty($rowData[self::COLUMN_STORE]) && !isset($this-
>_storeCodeToId[$rowData[self::COLUMN_STORE]])) {
                $this->addRowError(self::ERROR_INVALID_STORE,
$rowNum);
            }
            if (isset($rowData['password']) &&
strlen($rowData['password'])
                && $this->string->strlen($rowData['password']) <
self::MIN_PASSWORD_LENGTH
            ) {
                $this->addRowError(self::ERROR_PASSWORD_LENGTH,
$rowNum);
            }
            foreach ($this->_attributes as $attributeCode =>
$attributeData) {
                if (in_array($attributeCode, $this-
>_ignoredAttributes)) {
                    continue;
                }
                if (isset($rowData[$attributeCode]) &&
strlen($rowData[$attributeCode])) {
                    $this->isAttributeValid(
                        $attributeCode,
                        $attributeData,
                        $rowData,
                        $rowNum,
                        isset($this-
>_parameters[Import::FIELD_FIELD_MULTIPLE_VALUE_SEPARATOR])
                            ? $this-
>_parameters[Import::FIELD_FIELD_MULTIPLE_VALUE_SEPARATOR]
                            :
Import::DEFAULT_GLOBAL_MULTI_VALUE_SEPARATOR
                    );
                } elseif ($attributeData['is_required'] && !$this-
>_getCustomerId($email, $website)) {
                    $this->addRowError(self::ERROR_VALUE_IS_REQUIRED,
$rowNum, $attributeCode);
                }
            }
        }
    }
}
```

```
<?php
/**
 *
 * @author   : Ivan Studio <i1.1Kalyvan@gmail.com>
 */

namespace Ivan\ImportExport\Model;

use Exception;
use Ivan\ImportExport\Api\ExportJobRepositoryInterface;
use Ivan\ImportExport\Model\Export\Adapter\Factory as
FireExportAdapterFactory;
use Ivan\ImportExport\Model\Export\Dependencies\Config as
FireExportDiConfig;
use Ivan\ImportExport\Model\Export\EntityInterface;
use Ivan\ImportExport\Model\Source\Type\File\Config as
FireExportConfig;
use Ivan\ImportExport\Traits\General as GeneralTrait;
use Magento\Framework\App\Config\ScopeConfigInterface;
use Magento\Framework\Exception\LocalizedException;
use Magento\Framework\Filesystem;
use Magento\Framework\Phrase;
use Magento\Framework\Serialize\SerializerInterface;
use Magento\ImportExport\Model\Export as MagentoModelExport;
use Magento\ImportExport\Model\Export\Adapter\AbstractAdapter;
```

```
 use Magento\ImportExport\Model\Export\AbstractEntity as
AbstractEntity;
 use Magento\ImportExport\Model\Export\Entity\AbstractEntity as
EntityAbstractEntity;
 use Magento\ImportExport\Model\Export\Entity\Factory as
EntityFactory;
 use Magento\ImportExport\Model\Export\ConfigInterface as
ExportConfigInterface;
 use Psr\Log\LoggerInterface;
 use Symfony\Component\Console\Output\ConsoleOutput;

 /**
  * Class Export
  *
  * @package Ivan\ImportExport\Model
  */
 class Export extends MagentoModelExport
 {
     use GeneralTrait;

     /**
      * @var ScopeConfigInterface
      */
     protected $scopeConfig;

     /**
      * @var FireExportDiConfig
      */
     protected $fireExportDiConfig;

     /**
      * @var FireExportConfig
      */
     protected $fireExportConfig;

     /**
      * @var ExportJobRepositoryInterface
      */
     protected $exportJobRepository;

     /**
      * @var FireExportAdapterFactory
      */
     protected $_exportAdapterFac;

     /**
      * @var SerializerInterface
      */
     protected $jsonSerializer;

     /**
      * Export constructor.
      *
      * @param Filesystem $filesystem
      * @param ExportConfigInterface $exportConfig
      * @param EntityFactory $entityFactory
      * @param FireExportAdapterFactory $exportAdapterFac
      * @param ScopeConfigInterface $scopeConfig
      * @param FireExportDiConfig $fireExportDiConfig
      * @param FireExportConfig $fireExportConfig
      * @param ExportJobRepositoryInterface $exportJobRepository
      * @param SerializerInterface $serializer
      * @param LoggerInterface $logger
      * @param ConsoleOutput $output
      * @param array $data
      */
     public function __construct(
         Filesystem $filesystem,
         ExportConfigInterface $exportConfig,
         EntityFactory $entityFactory,
         FireExportAdapterFactory $exportAdapterFac,
         ScopeConfigInterface $scopeConfig,
         FireExportDiConfig $fireExportDiConfig,
         FireExportConfig $fireExportConfig,
         ExportJobRepositoryInterface $exportJobRepository,
         SerializerInterface $serializer,
         LoggerInterface $logger,
         ConsoleOutput $output,
         array $data = []
     ) {
         $this->scopeConfig = $scopeConfig;
         $this->fireExportDiConfig = $fireExportDiConfig;
         $this->fireExportConfig = $fireExportConfig;
         $this->exportJobRepository = $exportJobRepository;
         $this->jsonSerializer = $serializer;
         $this->output = $output;

         parent::__construct(
             $logger,
             $filesystem,
             $exportConfig,
             $entityFactory,
             $exportAdapterFac,
             $data
         );
     }

     /**
      * Add log comment
      *
      * @param mixed $debugData
      * @return $this
      */
     public function addLogComment($debugData)
     {
         if (is_array($debugData)) {
             $this->_logTrace = array_merge($this->_logTrace,
$debugData);
         } else {
             $this->_logTrace[] = $debugData;
         }

         if (is_scalar($debugData)) {
             $this->addLogWriteln($debugData, null, 'debug');
         } else {
             foreach ($debugData as $message) {
                 if ($message instanceof Phrase) {
                     $this->addLogWriteln($message->__toString(),
null, 'debug');
                 } else {
                     $this->addLogWriteln($message, null, 'debug');
                 }
             }
         }

         return $this;
     }

     /**
      * Retrieve Entity Model
      *
      * @return EntityInterface
      * @throws LocalizedException
      */
     protected function _getEntityAdapter()
     {
         if (!$this->_entityAdapter) {
             $entities = $this->fireExportDiConfig->get();
             if (isset($entities[$this->getEntity()])) {
                 $entity = $entities[$this->getEntity()];
                 try {
                     $this->_entityAdapter = $this->_entityFactory-
>create($entity['model']);
                 } catch (Exception $e) {
                     $this->_logger->critical($e);
                     $this->addLogWriteln($e->getMessage(), $this-
>output, 'error');
                     throw new LocalizedException(__('Please enter a
correct entity model.'));
                 }
                 if (!$this->_entityAdapter instanceof
EntityInterface) {
                     throw new LocalizedException(
                         __('The entity adapter object must be an
instance of %1.', EntityInterface::class)
                     );
                 }

                 if (!$this->_entityAdapter instanceof
EntityAbstractEntity
                     && !$this->_entityAdapter instanceof
AbstractEntity
                 ) {
                     throw new LocalizedException(
                         __(
                             'The entity adapter object must be an
instance of %1 or %2.',
                             EntityAbstractEntity::class,
                             AbstractEntity::class
                         )
                     );
                 }

                 if ($this->getEntity() != $this->_entityAdapter-
>getEntityTypeCode()) {
                     throw new LocalizedException(__('The input entity
code is not equal to entity adapter code.'));
                 }

                 $data = $this->getData();
                 if (empty($data['behavior_data']['deps']) &&
isset($entity['fields'])) {
                     $data['behavior_data']['deps'] =
array_keys($entity['fields']);
                 }
                 $this->_entityAdapter->setParameters($data);
             } else {
                 throw new LocalizedException(__('Please enter a
correct entity.'));
             }
         }

         return $this->_entityAdapter;
     }

     /**
      * Export data.
      *
      * @return string
      * @throws LocalizedException
      */
     public function export()
     {
         if (isset($this->_data[self::FILTER_ELEMENT_GROUP])) {
             $this->addLogComment(__('Begin export of %1', $this-
>getEntity()));

             $countRows = 0;
             $lastEntityId = 0;
             $exportData = $this->_getEntityAdapter()
```

```
                ->setLogger($this->_logger)
                ->setWriter($this->_getWriter())
                ->export();
            $result = $exportData[0];
            if (isset($exportData[1])) {
                $countRows = (int)$exportData[1];
            }
            if (isset($exportData[2])) {
                $lastEntityId = (int)$exportData[2];
            }
            $exportJob = $this->exportJobRepository->getById($this-
>getData('job_id'));
            $sourceData = $this->jsonSerializer-
>unserialize($exportJob->getExportSource());
            if ($lastEntityId > 0) {
                $sourceData = array_merge($sourceData,
['last_entity_id' => $lastEntityId]);
                $sourceData = $this->jsonSerializer-
>serialize($sourceData);
                $exportJob->setExportSource($sourceData);
                $this->exportJobRepository->save($exportJob);
            }
            if (!$countRows) {
                $this->addLogComment([__('There is no data for the
export.')]);

                return false;
            }
            if ($result) {
                $this->addLogComment([__('Exported %1 items.',
$countRows), __('The export is finished.')]);
            }
            return $result;
        } else {
            throw new LocalizedException(__('Please provide filter
data.'));
        }
    }

    /**
     * Retrieve Writer
     *
     * @return mixed
     * @throws LocalizedException
     */
    protected function _getWriter()
    {
        if (!$this->_writer) {
            $data = $this->fireExportConfig->get();
            $fileFormats = $data['export'];
            if (isset($fileFormats[$this->getFileFormat()])) {
                try {
                    $this->_writer = $this->_exportAdapterFac-
>create(
                        $fileFormats[$this-
>getFileFormat()]['model'],
                        ['data' => $this->_data]
                    );
                } catch (Exception $e) {
                    $this->_logger->critical($e);
                    throw new LocalizedException(__('Please enter a
correct entity model.'));
                }
                if (!$this->_writer instanceof AbstractAdapter) {
                    throw new LocalizedException(
                        __('The adapter object must be an instance of
%1.', AbstractAdapter::class)
                    );
                }
            } else {
                throw new LocalizedException(__('Please correct the
file format.'));
            }
        }
        return $this->_writer;
    }

    /**
     * Retrieve entity field for export
     *
     * @return array
     * @throws LocalizedException
     */
    public function getFields()
    {
        return $this->_getEntityAdapter()->getFieldsForExport();
    }
}
```

```
<?php
/**
 *
 * @author   : Ivan Studio <i1.1Kalyvan@gmail.com>
 */

namespace Ivan\ImportExport\Model;

use Magento\Framework\Model\AbstractModel;
use Ivan\ImportExport\Api\Data\ExportInterface;
use Ivan\ImportExport\Model\ResourceModel\ExportJob as
ResourceModelExportJob;

/**
 * Class ExportJob
 *
 * @package Ivan\ImportExport\Model
 */
class ExportJob extends AbstractModel implements ExportInterface
{
    protected function _construct()
    {
        $this->_init(ResourceModelExportJob::class);
    }

    /**
     * @return int|null
     */
    public function getId()
    {
        return $this->getData(self::ENTITY_ID);
    }

    /**
     * @return string
     */
    public function getTitle()
    {
        return $this->getData(self::TITLE);
    }

    /**
     * @return int
     */
    public function getIsActive()
    {
        return $this->getData(self::IS_ACTIVE);
    }

    /**
     * @return string|null
     */
    public function getCron()
    {
        return $this->getData(self::CRON);
    }

    /**
     * @return string
     */
    public function getFrequency()
    {
        return $this->getData(self::FREQUENCY);
    }

    /**
     * @return string
     */
    public function getEntity()
    {
        return $this->getData(self::ENTITY);
    }

    /**
     * @return string
     */
    public function getBehaviorData()
    {
        return  $this->getData(self::BEHAVIOR_DATA);
    }

    /**
     * @return string
     */
    public function getSourceData()
    {
        return $this->getData(self::SOURCE_DATA);
    }

    /**
     * @return string|null
     */
    public function getFileUpdatedAt()
    {
        return $this->getData(self::FILE_UPDATED_AT);
    }

    /**
     * @return string
     */
    public function getExportSource()
    {
        return $this->getData(self::EXPORT_SOURCE);
    }

    /**
     * @return string
     */
    public function getXslt()
    {
        return $this->getData(self::XSLT);
    }

    /**
     * @param $jobId
     *
     * @return ExportJob
     */
    public function setId($jobId)
    {
        $this->setData(self::ENTITY_ID, $jobId);
```

```
    }

    /**
     * @param $title
     *
     * @return ExportJob
     */
    public function setTitle($title)
    {
        $this->setData(self::TITLE, $title);
    }

    /**
     * @param $isActive
     *
     * @return ExportJob
     */
    public function setIsActive($isActive)
    {
        $this->setData(self::IS_ACTIVE, $isActive);
    }

    /**
     * @param $cron
     *
     * @return ExportJob
     */
    public function setCron($cron)
    {
        $this->setData(self::CRON, $cron);
    }

    /**
     * @param $frequency
     *
     * @return ExportJob
     */
    public function setFrequency($frequency)
    {
        $this->setData(self::FREQUENCY, $frequency);
    }

    /**
     * @param $entity
     *
     * @return ExportJob
     */
    public function setEntity($entity)
    {
        $this->setData(self::ENTITY, $entity);
    }

    /**
     * @param $behavior
     *
     * @return ExportJob
     */
    public function setBehaviorData($behavior)
    {
        $this->setData(self::BEHAVIOR_DATA, $behavior);
    }

    /**
     * @param $source
     *
     * @return ExportJob
     */
    public function setSourceData($source)
    {
        $this->setData(self::SOURCE_DATA, $source);
    }

    /**
     * @param $date
     *
     * @return ExportJob
     */
    public function setFileUpdatedAt($date)
    {
        $this->setData(self::FILE_UPDATED_AT, $date);
    }

    /**
     * @param $source
     *
     * @return ExportJob
     */
    public function setExportSource($source)
    {
        $this->setData(self::EXPORT_SOURCE, $source);
    }

    /**
     * @param $xslt
     *
     * @return ExportJob
     */
    public function setXslt($xslt)
    {
        return $this->setData(self::XSLT, $xslt);
    }
}
```

```
<?php
/**
 * GmbH.
 * @author   : Ivan Studio <i1.1Kalyvan@gmail.com>
 */

namespace Ivan\ImportExport\Model;

use Magento\Framework\Exception\CouldNotDeleteException;
use Magento\Framework\Exception\CouldNotSaveException;
use Magento\Framework\Exception\NoSuchEntityException;
use Ivan\ImportExport\Api\ExportJobRepositoryInterface;
use Ivan\ImportExport\Model\ResourceModel\ExportJob as
ExportJobResource;
use Ivan\ImportExport\Model\ExportJobFactory;
use Ivan\ImportExport\Model\ResourceModel\ExportJob\CollectionFactory
as ExportCollectionFactory;

/**
 * Class ExportJobRepository
 *
 * @package Ivan\ImportExport\Model
 */
class ExportJobRepository implements ExportJobRepositoryInterface
{
    /**
     * @var ExportJobResource
     */
    protected $resource;

    /**
     * @var \Ivan\ImportExport\Model\ExportJobFactory
     */
    protected $exportFactory;

    /**
     * @var ExportCollectionFactory
     */
    protected $exportCollectionFactory;

    /**
     * ExportJobRepository constructor.
     *
     * @param ExportJobResource                              $resource
     * @param \Ivan\ImportExport\Model\ExportJobFactory
$exportFactory
     * @param
ExportCollectionFactory                       $exportCollectionFactory
     */
    public function __construct(
        ExportJobResource $resource,
        ExportJobFactory $exportFactory,
        ExportCollectionFactory $exportCollectionFactory
    ) {
        $this->resource                = $resource;
        $this->exportFactory           = $exportFactory;
        $this->exportCollectionFactory = $exportCollectionFactory;
    }

    /**
     * @param \Ivan\ImportExport\Api\Data\ExportInterface $job
     *
     * @return \Ivan\ImportExport\Api\Data\ExportInterface
     * @throws CouldNotSaveException
     */
    public function save(\Ivan\ImportExport\Api\Data\ExportInterface
$job)
    {
        try {
            $this->resource->save($job);
        } catch (\Exception $exception) {
            throw new CouldNotSaveException(
                __(
                    'Could not save the job: %1',
                    $exception->getMessage()
                )
            );
        }

        return $job;
    }

    /**
     * @param $jobId
     *
     * @return mixed
     * @throws NoSuchEntityException
     */
    public function getById($jobId)
    {
        $job = $this->exportFactory->create();
        $this->resource->load($job, $jobId);
        if (!$job->getId()) {
            throw new NoSuchEntityException(__('Job with id "%1" does
not exist.', $jobId));
        }

        return $job;
    }

    /**
     * @param \Ivan\ImportExport\Api\Data\ExportInterface $job
     *
     * @return bool
     * @throws CouldNotDeleteException
     */
```

```
    public function
delete(\Ivan\ImportExport\Api\Data\ExportInterface $job)
    {
        try {
            $this->resource->delete($job);
        } catch (\Exception $exception) {
            throw new CouldNotDeleteException(
                __(
                    'Could not delete the job: %1',
                    $exception->getMessage()
                )
            );
        }

        return true;
    }

    /**
     * @param $jobId
     *
     * @return bool
     * @throws CouldNotDeleteException
     * @throws NoSuchEntityException
     */
    public function deleteById($jobId)
    {
        return $this->delete($this->getById($jobId));
    }
}
```

```
<?php
namespace Ivan\ImportExport\Model\Import;

use Magento\Framework\Exception\FileSystemException;
use Ivan\ImportExport\Api\Data\ImportInterface;
use Ivan\ImportExport\Traits\Import\Entity as ImportTrait;
use InvalidArgumentException;
use Zend_Validate_Exception;
use Magento\Framework\App\Config\ScopeConfigInterface;
use Magento\Framework\App\ResourceConnection;
use Magento\Framework\Exception\LocalizedException;
use Magento\ImportExport\Model\Import;
use Magento\ImportExport\Model\Import\AbstractEntity;
use Magento\ImportExport\Model\Import\AbstractSource;
use Magento\ImportExport\Model\ImportFactory;
use Zend\Serializer\Serializer;
use Ivan\ImportExport\Model\Source\Import\Behavior\Jobs as
JobsBehavior;
use Ivan\ImportExport\Api\JobRepositoryInterface;
use Ivan\ImportExport\Model\Job\Converter as JobConverter;
use Ivan\ImportExport\Model\ResourceModel\Job\CollectionFactory as
JobCollectionFactory;

/**
 * Class ImportJobs
 * @package Ivan\ImportExport\Model\Import
 */
class ImportJobs extends AbstractEntity implements
ImportAdapterInterface
{
    use ImportTrait;

    /**
     * Entity Type Code
     */
    const ENTITY_TYPE_CODE = 'import_jobs';

    /**
     * @var ResourceConnection
     */
    protected $_resource;

    /**
     * @var array
     */
    private $_alreadyDeleted = [];

    /**
     * @var JobRepositoryInterface
     */
    protected $jobRepository;

    /**
     * @var JobConverter
     */
    protected $jobConverter;

    /**
     * @var JobCollectionFactory
     */
    protected $jobCollectionFactory;

    /**
     * @param Context $context
     * @param ScopeConfigInterface $scopeConfig
     * @param ImportFactory $importFactory
     * @param JobRepositoryInterface $jobRepository
     * @param JobConverter $jobConverter
     * @param JobCollectionFactory $jobCollectionFactory
     * @param array $data
     */
    public function __construct(
        Context $context,
        ScopeConfigInterface $scopeConfig,
        ImportFactory $importFactory,
        JobRepositoryInterface $jobRepository,
        JobConverter $jobConverter,
        JobCollectionFactory $jobCollectionFactory,
        array $data = []
    ) {
        parent::__construct(
            $context->getStringUtils(),
            $scopeConfig,
            $importFactory,
            $context->getResourceHelper(),
            $context->getResource(),
            $context->getErrorAggregator(),
            $data
        );

        $this->output = $context->getOutput();
        $this->_resource = $context->getResource();
        $this->_importExportData = $context->getImportExportData();
        $this->_resourceHelper = $context->getResourceHelper();
        $this->jsonHelper = $context->getJsonHelper();
        $this->_dataSourceModel = $context->getDataSourceModel();
        $this->jobConverter = $jobConverter;
        $this->jobCollectionFactory = $jobCollectionFactory;
        $this->jobRepository = $jobRepository;
    }

    /**
     * Source model setter
     *
     * @param AbstractSource $source
     * @return $this
     */
    public function setSource(AbstractSource $source)
    {
        $this->_source = $source;
        $this->_dataValidated = false;

        return $this;
    }

    /**
     * Inner source object getter
     *
     * @return AbstractSource
     * @throws LocalizedException
     */
    protected function _getSource()
    {
        if (!$this->_source) {
            throw new LocalizedException(__('Please specify a
source.'));
        }

        return $this->_source;
    }

    /**
     * Import Behavior Getter
     *
     * @param array $rowData
     * @return string
     */
    public function getBehavior(array $rowData = null)
    {
        if (isset($this->_parameters['behavior'])) {
            return $this->_parameters['behavior'];
        }

        return Import::getDefaultBehavior();
    }

    /**
     * Import Data Rows
     *
     * @return boolean
     * @throws FileSystemException
     * @throws LocalizedException
     * @throws Zend_Validate_Exception
     */
    protected function _importData()
    {
        while ($bunch = $this->_dataSourceModel->getNextBunch()) {
            foreach ($bunch as $rowNumber => $rowData) {
                /* Validate data */
                if (!$rowData || !$this->validateRow($rowData,
$rowNumber)) {
                    continue;
                }

                if ($this->getErrorAggregator()->hasToBeTerminated())
{
                    $this->getErrorAggregator()-
>addRowToSkip($rowNumber);
                    continue;
                }

                /* Behavior selector */
                switch ($this->getBehavior()) {
                    case Import::BEHAVIOR_DELETE:
                        $this->_deleteJob($rowData);
                        break;
                    case Import::BEHAVIOR_REPLACE:
                        $this->_saveJob(
                            $this->_prepareDataForReplace($rowData)
                        );
```

```
                        break;
                    case Import::BEHAVIOR_APPEND:
                        $this->_saveJob(
                            $this->_prepareDataForUpdate($rowData)
                        );
                        break;
                    case JobsBehavior::ONLY_ADD:
                        $this->_saveJob(
                            $this->_prepareDataForOnlyAdd($rowData)
                        );
                        break;
                    case JobsBehavior::ONLY_UPDATE:
                        $this->_saveJob(
                            $this-
>_prepareDataForOnlyUpdate($rowData)
                        );
                        break;
                }
            }
        }

        return true;
    }

    /**
     * Save Job
     *
     * @param array $rowData
     * @throws LocalizedException
     */
    protected function _saveJob(array $rowData)
    {
        if ($rowData) {
            $this->jobRepository->save(
                $this->jobConverter->getDataObjectByData($rowData)
            );
        }
    }

    /**
     * Delete Import Job
     *
     * @param array $rowData
     * @throws LocalizedException
     */
    protected function _deleteJob(array $rowData)
    {
        $entityId = $rowData[ImportInterface::ENTITY_ID];
        if (!in_array($entityId, $this->_alreadyDeleted)) {
            $time = explode(" ", microtime());
            $startTime = $time[0] + $time[1];
            try {
                $this->jobRepository->deleteById($entityId);

                $time = explode(" ", microtime());
                $endTime = $time[0] + $time[1];
                $totalTime = $endTime - $startTime;
                $totalTime = round($totalTime, 5);
                $message = 'deleted job with entityId: %1 .... %2s';

                $this->addLogWriteln(__($message, $entityId,
$totalTime), $this->output, 'info');
            } catch (\Exception $exception) {
                $message = 'Job with entityId %1 not exist';
                $this->addLogWriteln(__($message, $entityId), $this-
>output, 'info');
            }

            $this->_alreadyDeleted[] = $entityId;
        }
    }

    /**
     * Validate Data Row
     *
     * @param array $rowData
     * @param int $rowNumber
     * @return boolean
     * @throws LocalizedException
     * @throws Zend_Validate_Exception
     */
    public function validateRow(array $rowData, $rowNumber)
    {
        if (isset($this->_validatedRows[$rowNumber])) {
            /* Check that row is already validated */
            return !$this->getErrorAggregator()-
>isRowInvalid($rowNumber);
        }

        $this->_validatedRows[$rowNumber] = true;
        $this->_processedEntitiesCount++;

        /* Behavior selector */
        switch ($this->getBehavior()) {
            case Import::BEHAVIOR_DELETE:
                $this->_validateRowForDelete($rowData, $rowNumber);
                break;
            case Import::BEHAVIOR_REPLACE:
                $this->_validateRowForReplace($rowData, $rowNumber);
                break;
            case Import::BEHAVIOR_ADD_UPDATE:
                $this->_validateRowForUpdate($rowData, $rowNumber);
                break;
            case JobsBehavior::ONLY_ADD:
                $this->_validateRowForOnlyAdd($rowData, $rowNumber);
                break;
            case JobsBehavior::ONLY_UPDATE:
                $this->_validateRowForOnlyUpdate($rowData,
$rowNumber);
                break;
        }

        return !$this->getErrorAggregator()-
>isRowInvalid($rowNumber);
    }

    /**
     * Validate Row Data For Update Behaviour
     *
     * @param array $rowData
     * @param $rowNumber
     */
    protected function _validateRowForUpdate(array $rowData,
$rowNumber)
    {
        if (!isset($rowData[ImportInterface::ENTITY_ID]) ||
empty($rowData[ImportInterface::ENTITY_ID])) {
            $this->addRowError('Invalid row %1', $rowNumber);
        }
    }

    /**
     * Validate Row Data For Only Add Behaviour
     *
     * @param array $rowData
     * @param $rowNumber
     */
    protected function _validateRowForOnlyAdd(array $rowData,
$rowNumber)
    {
        if (!isset($rowData[ImportInterface::ENTITY_ID]) ||
empty($rowData[ImportInterface::ENTITY_ID])) {
            $this->addRowError('Invalid row %1', $rowNumber);
        }
    }

    /**
     * Validate Row Data For Replace Behaviour
     *
     * @param array $rowData
     * @param $rowNumber
     */
    protected function _validateRowForReplace(array $rowData,
$rowNumber)
    {
        if (!isset($rowData[ImportInterface::ENTITY_ID]) ||
empty($rowData[ImportInterface::ENTITY_ID])) {
            $this->addRowError('Invalid row %1', $rowNumber);
        }
    }

    /**
     * Validate Row Data For Delete Behaviour
     *
     * @param array $rowData
     * @param $rowNumber
     */
    protected function _validateRowForDelete(array $rowData,
$rowNumber)
    {
        if (!isset($rowData[ImportInterface::ENTITY_ID]) ||
empty($rowData[ImportInterface::ENTITY_ID])) {
            $this->addRowError('Invalid row %1', $rowNumber);
        }
    }

    /**
     * Validate Row Data For Only Update Behaviour
     *
     * @param array $rowData
     * @param $rowNumber
     */
    protected function _validateRowForOnlyUpdate(array $rowData,
$rowNumber)
    {
        if (!isset($rowData[ImportInterface::ENTITY_ID]) ||
empty($rowData[ImportInterface::ENTITY_ID])) {
            $this->addRowError('Invalid row %1', $rowNumber);
        }
    }

    /**
     * Retrieve Entity Type Code
     *
     * @return string
     */
    public function getEntityTypeCode()
    {
        return self::ENTITY_TYPE_CODE;
    }

    /**
     * Prepare Data For Update
     *
     * @param array $rowData
     * @return array
     * @throws FileSystemException
     * @throws LocalizedException
     */
    private function _prepareDataForUpdate(array $rowData)
```

```
    {
        if (!$this->_prepareDataForOnlyUpdate($rowData)) {
            unset($rowData[ImportInterface::ENTITY_ID]);
        }

        return $rowData;
    }

    /**
     * Prepare Data For Only Update
     *
     * @param array $rowData
     * @return array
     * @throws FileSystemException
     * @throws LocalizedException
     */
    private function _prepareDataForOnlyAdd(array $rowData)
    {
        try {
            $this->jobRepository-
>getById($rowData[ImportInterface::ENTITY_ID]);
            return null;
        } catch (\Exception $exception) {
            unset($rowData[ImportInterface::ENTITY_ID]);
            $this->addLogWriteln(
                __($exception->getMessage(),
$rowData[ImportInterface::ENTITY_ID]),
                $this->output,
                'info'
            );
        }

        return $rowData;
    }

    /**
     * Prepare Data For Only Update
     *
     * @param array $rowData
     * @return array
     * @throws FileSystemException
     * @throws LocalizedException
     */
    private function _prepareDataForOnlyUpdate(array $rowData)
    {
        try {
            $this->jobRepository-
>getById($rowData[ImportInterface::ENTITY_ID]);
            unset($rowData[ImportInterface::ENTITY_ID]);
            return $rowData;
        } catch (\Exception $exception) {
            return null;
        }
    }

    /**
     * Prepare Data For Replace
     *
     * @param array $rowData
     * @return array
     * @throws FileSystemException
     * @throws LocalizedException
     */
    private function _prepareDataForReplace(array $rowData)
    {
        $this->_deleteJob($rowData);

        return $this->_prepareDataForUpdate($rowData);
    }

    /**
     * Save Validated Bunches
     *
     * @return $this
     * @throws \Magento\Framework\Exception\LocalizedException
     * @throws \Zend_Validate_Exception
     */
    protected function _saveValidatedBunches()
    {
        $source = $this->_getSource();
        $currentDataSize = 0;
        $bunchRows = [];
        $startNewBunch = false;
        $nextRowBackup = [];
        $maxDataSize = $this->_resourceHelper->getMaxDataSize();
        $bunchSize = $this->_importExportData->getBunchSize();

        $source->rewind();
        $this->_dataSourceModel->cleanBunches();
        $file = null;
        $jobId = null;

        if (isset($this->_parameters['file'])) {
            $file = $this->_parameters['file'];
        }
        if (isset($this->_parameters['job_id'])) {
            $jobId = $this->_parameters['job_id'];
        }

        while ($source->valid() || $bunchRows) {
            if ($startNewBunch || !$source->valid()) {
                $this->_dataSourceModel->saveBunches(
                    $this->getEntityTypeCode(),
                    $this->getBehavior(),
                    $jobId,
                    $file,
                    $bunchRows
                );

                $bunchRows = $nextRowBackup;
                $currentDataSize =
strlen(Serializer::serialize($bunchRows));
                $startNewBunch = false;
                $nextRowBackup = [];
            }

            if ($source->valid()) {
                try {
                    $rowData = $source->current();
                } catch (InvalidArgumentException $e) {
                    $this->addRowError($e->getMessage(), $this-
>_processedRowsCount);
                    $this->_processedRowsCount++;
                    $source->next();
                    continue;
                }
                $rowData = $this->customBunchesData($rowData);
                $this->_processedRowsCount++;
                if ($this->validateRow($rowData, $source->key())) {
                    $rowSize = strlen($this->jsonHelper-
>jsonEncode($rowData));
                    $isBunchSizeExceeded = $bunchSize > 0 &&
count($bunchRows) >= $bunchSize;

                    if ($currentDataSize + $rowSize >= $maxDataSize
|| $isBunchSizeExceeded) {
                        $startNewBunch = true;
                        $nextRowBackup = [$source->key() =>
$rowData];
                    } else {
                        $bunchRows[$source->key()] = $rowData;
                        $currentDataSize += $rowSize;
                    }
                }

                $source->next();
            }
        }

        return $this;
    }

    /**
     * Retrieve All Fields Source
     *
     * @return array
     */
    public function getAllFields()
    {
        return [];
    }
}
```

```
<?php
/**
 * @author   : Ivan Studio <i1.1Kalyvan@gmail.com>
 */

namespace Ivan\ImportExport\Model;

use Magento\Framework\ObjectManagerInterface;
use Magento\Framework\Module\Manager as ModuleManager;

/**
 * Class IntegrationFactory
 *
 * @package Ivan\ImportExport\Model
 */
class IntegrationFactory
{
    /**
     * @var \Magento\Framework\ObjectManagerInterface
     */
    private $objectManager;

    /**
     * @var ModuleManager
     */
    private $moduleManager;

    /**
     * @param ObjectManagerInterface $objectManager
     * @param ModuleManager $moduleManager
     */
    public function __construct(
        ObjectManagerInterface $objectManager,
        ModuleManager $moduleManager
    ) {
        $this->objectManager = $objectManager;
        $this->moduleManager = $moduleManager;
    }

    /**
     * @param string $type
     * @param array $data
     * @return mixed
     */
    public function create($type, array $data = [])
    {
        if ($this->isModuleEnabled($type)) {
```

```
            return $this->objectManager->create($type, $data);
        }
        return false;
    }

    /**
     * @param $type
     * @return bool
     */
    private function isModuleEnabled($type)
    {
        $moduleName = implode('_', array_slice(explode('\\', $type),
0, 2));
        return $this->moduleManager->isEnabled($moduleName);
    }
}
```

```
<?php
/**
 *
 * @author   : Ivan Studio <i1.1Kalyvan@gmail.com>
 */

namespace Ivan\ImportExport\Model;

use Magento\Framework\Exception\NoSuchEntityException;
use Magento\Store\Model\StoreManagerInterface;

/**
 * Registry for \Magento\Customer\Model\Customer
 */
class JobRegistry
{
    const REGISTRY_SEPARATOR = ':';

    /**
     * @var JobFactory
     */
    private $jobFactory;

    /**
     * @var array
     */
    private $jobRegistryById = [];

    /**
     * @var \Magento\Store\Model\StoreManagerInterface
     */
    private $storeManager;

    /**
     * Constructor
     *
     * @param JobFactory $jobFactory
     * @param StoreManagerInterface $storeManager
     */
    public function __construct(
        JobFactory $jobFactory,
        StoreManagerInterface $storeManager
    ) {
        $this->jobFactory = $jobFactory;
        $this->storeManager = $storeManager;
    }

    /**
     * Retrieve Job Model from registry given an id
     *
     * @param string $jobId
     * @return Job
     * @throws NoSuchEntityException
     */
    public function retrieve($jobId)
    {
        if (isset($this->jobRegistryById[$jobId])) {
            return $this->jobRegistryById[$jobId];
        }
        /** @var Customer $customer */
        $job = $this->jobFactory->create()->load($jobId);
        if (!$job->getId()) {
            // job does not exist
            throw NoSuchEntityException::singleField('jobId',
$jobId);
        } else {
            $this->jobRegistryById[$jobId] = $job;
            return $job;
        }
    }

    /**
     * Remove instance of the Job Model from registry given an id
     *
     * @param int $jobId
     * @return void
     */
    public function remove($jobId)
    {
        if (isset($this->jobRegistryById[$jobId])) {
            /** @var Customer $customer */
            unset($this->jobRegistryById[$jobId]);
        }
    }

    /**
     * Replace existing job model with a new one.
     *
     * @param Job $job
     * @return $this
     */
    public function push(Job $job)
    {
        $this->jobRegistryById[$job->getId()] = $job;
        return $this;
    }
}
```

```
<?php
/**
 * GmbH.
 * @author   : Ivan Studio <i1.1Kalyvan@gmail.com>
 */

namespace Ivan\ImportExport\Model;

use Ivan\ImportExport\Api\Data\ImportInterface;
use Ivan\ImportExport\Api\JobRepositoryInterface;
use Ivan\ImportExport\Model\ResourceModel\Job as ImportJobResource;
use Ivan\ImportExport\Model\ResourceModel\Job\CollectionFactory as
ImportCollectionFactory;
use Magento\Framework\Exception\CouldNotDeleteException;
use Magento\Framework\Exception\CouldNotSaveException;
use Magento\Framework\Exception\NoSuchEntityException;

/**
 * Class JobRepository
 *
 * @package Ivan\ImportExport\Model
 */
class JobRepository implements JobRepositoryInterface
{

    /**
     * @var ImportJobResource
     */
    protected $resource;

    /**
     * @var JobFactory
     */
    protected $importFactory;

    /**
     * @var ImportCollectionFactory
     */
    protected $importCollectionFactory;

    /**
     * @var \Ivan\ImportExport\Model\JobRegistry
     */
    protected $jobRegistry;

    /**
     * JobRepository constructor.
     * @param ImportJobResource $resource
     * @param JobRegistry $jobRegistry
     * @param JobFactory $importFactory
     * @param ImportCollectionFactory $importCollectionFactory
     */
    public function __construct(
        ImportJobResource $resource,
        \Ivan\ImportExport\Model\JobRegistry $jobRegistry,
        \Ivan\ImportExport\Model\JobFactory $importFactory,
        ImportCollectionFactory $importCollectionFactory
    ) {
        $this->resource = $resource;
        $this->jobRegistry = $jobRegistry;
        $this->importFactory = $importFactory;
        $this->importCollectionFactory = $importCollectionFactory;
    }

    /**
     * {@inheritdoc}
     */
    public function save(ImportInterface $job)
    {
        try {
            $this->resource->save($job);
        } catch (\Exception $exception) {
            throw new CouldNotSaveException(__(
                'Could not save the job: %1',
                $exception->getMessage()
            ));
        }

        return $job;
    }

    /**
     * {@inheritdoc}
     */
    public function getById($jobId)
    {
        $job = $this->importFactory->create();
        $this->resource->load($job, $jobId);

        if (!$job->getId()) {
            throw new NoSuchEntityException(__('Job with id "%1" does
not exist.', $jobId));
        }

        return $job;
```

```
    }

    /**
     * {@inheritdoc}
     */
    public function delete(ImportInterface $job)
    {
        try {
            $this->resource->delete($job);
        } catch (\Exception $exception) {
            throw new CouldNotDeleteException(__(
                'Could not delete the job: %1',
                $exception->getMessage()
            ));
        }
        return true;
    }

    /**
     * {@inheritdoc}
     */
    public function deleteById($jobId)
    {
        return $this->delete($this->getById($jobId));
    }
}
```

```
<?php
namespace Ivan\ImportExport\Model\Import;

use Ivan\ImportExport\Traits\Import\Entity as ImportTrait;
use Magento\Framework\App\Config\ScopeConfigInterface;
use Magento\Framework\Encryption\Encryptor;
use Magento\Framework\Encryption\EncryptorInterface;
use Magento\Customer\Model\CustomerFactory;
use Magento\ImportExport\Model\Import\AbstractEntity;
use Magento\ImportExport\Model\ImportFactory;
use Magento\ImportExport\Model\Import;
use Magento\Store\Model\StoreManagerInterface;
use Magento\Newsletter\Model\SubscriberFactory;
use Magento\Newsletter\Model\Subscriber;

/**
 * Newsletter Subscriber Import
 */
class NewsletterSubscriber extends AbstractEntity implements
ImportAdapterInterface
{
    use ImportTrait;

    /**
     * Subscriber id column name
     */
    const COL_SUBSCRIBER_ID = 'subscriber_id';

    /**
     * Subscriber email column name
     */
    const COL_SUBSCRIBER_EMAIL = 'subscriber_email';

    /**
     * Store id column name
     */
    const COL_STORE_ID = 'store_id';

    /**
     * Subscriber status column name
     */
    const COL_STATUS = 'subscriber_status';

    /**
     * Customer id column name
     */
    const COL_CUSTOMER_ID = 'customer_id';

    /**
     * Password hash column name
     */
    const COL_PASSWORD_HASH = 'password_hash';

    /**
     * Error codes
     */
    const ERROR_SUBSCRIBER_ID_IS_EMPTY = 'subscriberIdIsEmpty';
    const ERROR_STORE_ID_IS_EMPTY = 'storeIdIsEmpty';
    const ERROR_EMAIL_FORMAT = 'emailFormatInvalid';
    const ERROR_STATUS_VALUE = 'statusInvalid';

    /**
     * Validation failure message template definitions
     *
     * @var array
     */
    protected $_messageTemplates = [
        self::ERROR_SUBSCRIBER_ID_IS_EMPTY => 'Subscriber id is
empty',
        self::ERROR_STORE_ID_IS_EMPTY => 'Store id is empty',
        self::ERROR_EMAIL_FORMAT => 'Email has a wrong format',
        self::ERROR_STATUS_VALUE => 'Unsupported subscriber status
value',
    ];

    /**
     * Field list
     *
     * @var array
     */
    protected $fields = [
        'subscriber_id',
        'store_id',
        'change_status_at',
        'subscriber_email',
        'firstname',
        'lastname',
        'password_hash',
        'subscriber_status',
        'subscriber_confirm_code'
    ];

    /**
     * Subscriber factory
     *
     * @var SubscriberFactory
     */
    protected $subscriberFactory;

    /**
     * Store manager
     *
     * @var StoreManagerInterface
     */
    protected $storeManager;

    /**
     * Customer factory
     *
     * @var CustomerFactory
     */
    protected $customerFactory;

    /**
     * Encryptor
     *
     * @var EncryptorInterface
     */
    protected $encryptor;

    /**
     * Statuse codes
     *
     * @var array
     */
    protected $status = [
        Subscriber::STATUS_SUBSCRIBED,
        Subscriber::STATUS_NOT_ACTIVE,
        Subscriber::STATUS_UNSUBSCRIBED,
        Subscriber::STATUS_UNCONFIRMED
    ];

    /**
     * Initialize import
     *
     * @param Context $context
     * @param ScopeConfigInterface $scopeConfig
     * @param ImportFactory $importFactory
     * @param SubscriberFactory $subscriberFactory
     * @param CustomerFactory $customerFactory
     * @param StoreManagerInterface $storeManager
     * @param EncryptorInterface $encryptor
     * @param array $data
     */
    public function __construct(
        Context $context,
        ScopeConfigInterface $scopeConfig,
        ImportFactory $importFactory,
        SubscriberFactory $subscriberFactory,
        CustomerFactory $customerFactory,
        StoreManagerInterface $storeManager,
        EncryptorInterface $encryptor,
        array $data = []
    ) {
        $this->_logger = $context->getLogger();
        $this->output = $context->getOutput();
        $this->_importExportData = $context->getImportExportData();
        $this->_resourceHelper = $context->getResourceHelper();
        $this->jsonHelper = $context->getJsonHelper();
        $this->subscriberFactory = $subscriberFactory;
        $this->customerFactory = $customerFactory;
        $this->storeManager = $storeManager;
        $this->encryptor = $encryptor;

        parent::__construct(
            $context->getStringUtils(),
            $scopeConfig,
            $importFactory,
            $context->getResourceHelper(),
            $context->getResource(),
            $context->getErrorAggregator(),
            $data
        );
    }

    /**
     * Import data rows
     *
     * @return boolean
     */
    protected function _importData()
    {
        while ($bunch = $this->_dataSourceModel->getNextBunch()) {
```

```
            foreach ($bunch as $rowNumber => $rowData) {
                /* validate data */
                if (!$rowData || !$this->validateRow($rowData,
$rowNumber)) {
                    continue;
                }

                if ($this->getErrorAggregator()->hasToBeTerminated())
{
                    $this->getErrorAggregator()-
>addRowToSkip($rowNumber);
                    continue;
                }

                /* behavior selector */
                switch ($this->getBehavior()) {
                    case Import::BEHAVIOR_DELETE:
                        $this->delete($rowData);
                        break;
                    case Import::BEHAVIOR_REPLACE:
                        $this->delete($rowData);
                        $this->save($rowData);
                        break;
                    case Import::BEHAVIOR_ADD_UPDATE:
                        $this->save($rowData);
                        break;
                }
            }
        }
        return true;
    }

    /**
     * Imported entity type code getter
     *
     * @return string
     */
    public function getEntityTypeCode()
    {
        return 'newsletter_subscriber';
    }

    /**
     * Retrieve All Fields Source
     *
     * @return array
     */
    public function getAllFields()
    {
        return $this->fields;
    }

    /**
     * Validate data row
     *
     * @param array $rowData
     * @param int $rowNumber
     * @return bool
     */
    public function validateRow(array $rowData, $rowNumber)
    {
        if (isset($this->_validatedRows[$rowNumber])) {
            /* check that row is already validated */
            return !$this->getErrorAggregator()-
>isRowInvalid($rowNumber);
        }

        $this->_validatedRows[$rowNumber] = true;
        $this->_processedEntitiesCount++;

        /* behavior selector */
        switch ($this->getBehavior()) {
            case Import::BEHAVIOR_DELETE:
                $this->validateRowForDelete($rowData, $rowNumber);
                break;
            case Import::BEHAVIOR_REPLACE:
                $this->validateRowForReplace($rowData, $rowNumber);
                break;
            case Import::BEHAVIOR_ADD_UPDATE:
                $this->validateRowForUpdate($rowData, $rowNumber);
                break;
        }
        return !$this->getErrorAggregator()-
>isRowInvalid($rowNumber);
    }

    /**
     * Validate row data for replace behaviour
     *
     * @param array $rowData
     * @param int $rowNumber
     * @return void
     */
    public function validateRowForReplace(array $rowData, $rowNumber)
    {
        $this->validateRowForUpdate($rowData, $rowNumber);
    }

    /**
     * Validate row data for delete behaviour
     *
     * @param array $rowData
     * @param int $rowNumber
     * @return void
     */
    public function validateRowForDelete(array $rowData, $rowNumber)
    {
        if (!empty($rowData[self::COL_SUBSCRIBER_EMAIL])) {
            $this->validateStore($rowData, $rowNumber);
            $this->validateEmail($rowData, $rowNumber);
        } elseif (empty($rowData[self::COL_SUBSCRIBER_ID])) {
            $this->addRowError(self::ERROR_SUBSCRIBER_ID_IS_EMPTY,
$rowNumber);
        }
    }

    /**
     * Validate row data for update behaviour
     *
     * @param array $rowData
     * @param int $rowNumber
     * @return void
     */
    public function validateRowForUpdate(array $rowData, $rowNumber)
    {
        $this->validateRowForDelete($rowData, $rowNumber);
        $this->validateStatus($rowData, $rowNumber);
    }

    /**
     * Validate email string
     *
     * @param array $rowData
     * @param int $rowNumber
     * @return void
     */
    protected function validateEmail(array $rowData, $rowNumber)
    {
        if (!filter_var($rowData[self::COL_SUBSCRIBER_EMAIL],
FILTER_VALIDATE_EMAIL)) {
            $this->addRowError(self::ERROR_EMAIL_FORMAT, $rowNumber);
        }
    }

    /**
     * Validate store data
     *
     * @param array $rowData
     * @param int $rowNumber
     * @return void
     */
    protected function validateStore(array $rowData, $rowNumber)
    {
        if (!$this->storeManager->isSingleStoreMode()) {
            if (empty($rowData[self::COL_STORE_ID])) {
                $this->addRowError(self::ERROR_STORE_ID_IS_EMPTY,
$rowNumber);
            }
        }
    }

    /**
     * Validate status string
     *
     * @param array $rowData
     * @param int $rowNumber
     * @return void
     */
    protected function validateStatus(array $rowData, $rowNumber)
    {
        if (!empty($rowData[self::COL_STATUS])) {
            if (!in_array($rowData[self::COL_STORE_ID], $this-
>status)) {
                $this->addRowError(self::ERROR_STATUS_VALUE,
$rowNumber);
            }
        }
    }

    /**
     * Delete row
     *
     * @param array $rowData
     * @return $this
     */
    protected function delete(array $rowData)
    {
        $subscriber = $this->subscriberFactory->create();
        if (!empty($rowData[self::COL_SUBSCRIBER_EMAIL])) {
            $subscriber-
>loadByEmail($rowData[self::COL_SUBSCRIBER_EMAIL]);
        } elseif (!empty($rowData[self::COL_SUBSCRIBER_ID])) {
            $subscriber->load($rowData[self::COL_SUBSCRIBER_ID]);
        }

        if ($subscriber->getId()) {
            $subscriber->delete();
            $this->countItemsDeleted++;
        }
        return $this;
    }

    /**
     * Update entity
     *
     * @param array $rowData
     * @return $this
     */
    protected function save(array $rowData)
    {
```

```
        $subscriber = $this->subscriberFactory->create();
        $email = $rowData[self::COL_SUBSCRIBER_EMAIL] ?? null;

        if (!empty($email)) {
            $subscriber->loadByEmail($email);
            if ($subscriber->getId()) {
                unset($rowData[self::COL_SUBSCRIBER_EMAIL]);
            }
        } elseif (!empty($rowData[self::COL_SUBSCRIBER_ID])) {
            $subscriber->load($rowData[self::COL_SUBSCRIBER_ID]);
            unset($rowData[self::COL_SUBSCRIBER_ID]);
        }

        if ($subscriber->getId()) {
            $this->countItemsUpdated++;
        } else {
            $this->countItemsCreated++;
            $rowData[self::COL_SUBSCRIBER_ID] = null;
        }

        unset($rowData[self::COL_CUSTOMER_ID]);
        $subscriber->addData($rowData);

        if (!$subscriber->getCustomerId()) {
            $customer = $this->customerFactory->create();
            if ($customer->getSharingConfig()->isWebsiteScope()) {
                $store = empty($rowData[self::COL_STORE_ID])
                    ? $this->storeManager->getDefaultStoreView()
                    : $this->storeManager-
>getStore($rowData[self::COL_STORE_ID]);

                $customer->setWebsiteId($store->getWebsiteId());
            }

            $customer->loadByEmail($email);
            if ($customer->getId()) {
                $subscriber->setCustomerId($customer->getId());
            } else {
                if (!empty($rowData['firstname']) &&
                    !empty($rowData['lastname']) &&
                    !empty($rowData[self::COL_PASSWORD_HASH])
                ) {
                    $customer->setFirstname($rowData['firstname']);
                    $customer->setLastname($rowData['lastname']);
                    $customer->setEmail($subscriber-
>getSubscriberEmail());
                    $customer->setPasswordHash($this-
>getPasswordHash($rowData));
                    $customer->save();

                    $subscriber->setCustomerId($customer->getId());
                }
            }
        }

        try {
            $subscriber->save();
        } catch (\Exception $e) {
            $this->addLogWriteln($e->getMessage(), $this-
>getOutput(), 'error');

        }
        return $this;
    }

    /**
     * Retrieve customer password hash
     *
     * @param array $rowData
     * @return string
     */
    protected function getPasswordHash(array $rowData)
    {
        if ($this->encryptor-
>validateHashVersion($rowData[self::COL_PASSWORD_HASH])) {
            // m2 pasword hash
            return $rowData[self::COL_PASSWORD_HASH];
        }
        // m1 pasword hash
        $parts = explode(Encryptor::DELIMITER,
$rowData[self::COL_PASSWORD_HASH]);
        $count = count($parts);
        if (2 == $count) {
            list($hash, $salt) = $parts;
            $version = (32 == strlen($hash)) ?
Encryptor::HASH_VERSION_MD5 : Encryptor::HASH_VERSION_SHA256;
            return implode(
                Encryptor::DELIMITER,
                [
                    $hash,
                    $salt,
                    $version
                ]
            );
        }
        return null;
    }

    /**
     * Save Validated Bunches
     *
     * @return $this
     * @throws \Magento\Framework\Exception\LocalizedException
     */
    protected function _saveValidatedBunches()
    {
        $source = $this->getSource();
        $currentDataSize = 0;
        $bunchRows = [];
        $startNewBunch = false;
        $nextRowBackup = [];
        $maxDataSize = $this->_resourceHelper->getMaxDataSize();
        $bunchSize = $this->_importExportData->getBunchSize();

        $source->rewind();
        $this->_dataSourceModel->cleanBunches();
        $file = null;
        $jobId = null;
        if (isset($this->_parameters['file'])) {
            $file = $this->_parameters['file'];
        }
        if (isset($this->_parameters['job_id'])) {
            $jobId = $this->_parameters['job_id'];
        }

        while ($source->valid() || $bunchRows) {
            if ($startNewBunch || !$source->valid()) {
                $this->_dataSourceModel->saveBunches(
                    $this->getEntityTypeCode(),
                    $this->getBehavior(),
                    $jobId,
                    $file,
                    $bunchRows
                );
                $bunchRows = $nextRowBackup;
                $currentDataSize = strlen($this->jsonHelper-
>jsonEncode($bunchRows));
                $startNewBunch = false;
                $nextRowBackup = [];
            }

            if ($source->valid()) {
                try {
                    $rowData = $source->current();
                } catch (\InvalidArgumentException $e) {
                    $this->addRowError($e->getMessage(), $this-
>_processedRowsCount);
                    $this->_processedRowsCount++;
                    $source->next();
                    continue;
                }
                $rowData = $this->customBunchesData($rowData);
                $this->_processedRowsCount++;
                if ($this->validateRow($rowData, $source->key())) {
                    $rowSize = strlen($this->jsonHelper-
>jsonEncode($rowData));

                    $isBunchSizeExceeded = $bunchSize > 0 &&
count($bunchRows) >= $bunchSize;

                    if ($currentDataSize + $rowSize >= $maxDataSize
|| $isBunchSizeExceeded) {
                            $startNewBunch = true;
                            $nextRowBackup = [$source->key() =>
$rowData];
                    } else {
                            $bunchRows[$source->key()] = $rowData;
                            $currentDataSize += $rowSize;
                    }
                }
                $source->next();
            }
        }
        return $this;
    }
}
```

```
<?php

namespace Ivan\ImportExport\Model\Import;

use Exception;
use Ivan\ImportExport\Helper\Data as Helper;
use Ivan\ImportExport\Model\Import\Order\AddressFactory;
use Ivan\ImportExport\Model\Import\Order\Creditmemo;
use Ivan\ImportExport\Model\Import\Order\Creditmemo\Comment;
use Ivan\ImportExport\Model\Import\Order\Creditmemo\CommentFactory as
CreditmemoCommentFactory;
use Ivan\ImportExport\Model\Import\Order\Creditmemo\ItemFactory as
CreditmemoItemFactory;
use Ivan\ImportExport\Model\Import\Order\Creditmemo\GeneratorFactory
as CreditmemoGeneratorFactory;
use Ivan\ImportExport\Model\Import\Order\CreditmemoFactory;
use Ivan\ImportExport\Model\Import\Order\DataProcessor;
use Ivan\ImportExport\Model\Import\Order\Entity;
use Ivan\ImportExport\Model\Import\Order\EntityFactory;
use Ivan\ImportExport\Model\Import\Order\Invoice;
use Ivan\ImportExport\Model\Import\Order\Invoice\CommentFactory as
InvoiceCommentFactory;
use Ivan\ImportExport\Model\Import\Order\Invoice\ItemFactory as
InvoiceItemFactory;
use Ivan\ImportExport\Model\Import\Order\InvoiceFactory;
use Ivan\ImportExport\Model\Import\Order\ItemFactory;
use Ivan\ImportExport\Model\Import\Order\Payment;
use Ivan\ImportExport\Model\Import\Order\Payment\Transaction;
use Ivan\ImportExport\Model\Import\Order\Payment\TransactionFactory;
use Ivan\ImportExport\Model\Import\Order\PaymentFactory;
use Ivan\ImportExport\Model\Import\Order\Shipment;
use Ivan\ImportExport\Model\Import\Order\Shipment\CommentFactory as
ShipmentCommentFactory;
```

```
 use Ivan\ImportExport\Model\Import\Order\Shipment\ItemFactory as
ShipmentItemFactory;
 use Ivan\ImportExport\Model\Import\Order\Shipment\Track;
 use Ivan\ImportExport\Model\Import\Order\Shipment\TrackFactory as
ShipmentTrackFactory;
 use Ivan\ImportExport\Model\Import\Order\ShipmentFactory;
 use Ivan\ImportExport\Model\Import\Order\Status\HistoryFactory as
StatusHistoryFactory;
 use Ivan\ImportExport\Model\Import\Order\Tax;
 use Ivan\ImportExport\Model\Import\Order\Tax\Item;
 use Ivan\ImportExport\Model\Import\Order\Tax\ItemFactory as
TaxItemFactory;
 use Ivan\ImportExport\Model\Import\Order\TaxFactory;
 use Ivan\ImportExport\Traits\Import\Entity as ImportTrait;
 use InvalidArgumentException;
 use Magento\Framework\Exception\LocalizedException;
 use Magento\Framework\Stdlib\DateTime\TimezoneInterface;
 use Magento\ImportExport\Model\Import\AbstractSource;
 use Magento\ImportExport\Model\Import\Entity\AbstractEntity;
 use
Magento\ImportExport\Model\Import\ErrorProcessing\ProcessingErrorAggre
gatorInterface;
 use Magento\OfflinePayments\Model\Checkmo;
 use Magento\Sales\Model\ResourceModel\GridPool;

 /**
  * Order Import
  */
 class Order extends AbstractEntity implements ImportAdapterInterface
 {
     use ImportTrait;

     /**
      * Entity Type Code
      *
      */
     const ENTITY_TYPE_CODE = 'order';

     /**
      * Order Entity Adapter
      *
      * @var Entity
      */
     protected $_order;

     /**
      * Item Entity Adapter
      *
      * @var \Ivan\ImportExport\Model\Import\Order\Item
      */
     protected $_item;

     /**
      * Address Entity Adapter
      *
      * @var \Ivan\ImportExport\Model\Import\Order\Address
      */
     protected $_address;

     /**
      * Shipment Entity Adapter
      *
      * @var Shipment
      */
     protected $_shipment;

     /**
      * Shipment Item Entity Adapter
      *
      * @var \Ivan\ImportExport\Model\Import\Order\Shipment\Item
      */
     protected $_shipmentItem;

     /**
      * Shipment Track Entity Adapter
      *
      * @var Track
      */
     protected $_shipmentTrack;

     /**
      * Shipment Comment Entity Adapter
      *
      * @var \Ivan\ImportExport\Model\Import\Order\Shipment\Comment
      */
     protected $_shipmentComment;

     /**
      * Payment Entity Adapter
      *
      * @var Payment
      */
     protected $_payment;

     /**
      * Payment Transaction Entity Adapter
      *
      * @var Transaction
      */
     protected $_transaction;

     /**
      * Invoice Entity Adapter
      *
      * @var Invoice
      */
     protected $_invoice;

     /**
      * Invoice Item Entity Adapter
      *
      * @var \Ivan\ImportExport\Model\Import\Order\Invoice\Item
      */
     protected $_invoiceItem;

     /**
      * Invoice Comment Entity Adapter
      *
      * @var \Ivan\ImportExport\Model\Import\Order\Invoice\Comment
      */
     protected $_invoiceComment;

     /**
      * Creditmemo Entity Adapter
      *
      * @var Creditmemo
      */
     protected $_creditmemo;

     /**
      * Creditmemo Item Entity Adapter
      *
      * @var \Ivan\ImportExport\Model\Import\Order\Creditmemo\Item
      */
     protected $_creditmemoItem;

     /**
      * Creditmemo Comment Entity Adapter
      *
      * @var Comment
      */
     protected $_creditmemoComment;

     /**
      * Creditmemo Comment Generator
      *
      * @var Comment
      */
     protected $_creditmemoGenerator;

     /**
      * Tax Entity Adapter
      *
      * @var Tax
      */
     protected $_tax;

     /**
      * Tax Item Adapter
      *
      * @var Item
      */
     protected $_taxItem;

     /**
      * Status History Entity Adapter
      *
      * @var \Ivan\ImportExport\Model\Import\Order\Status\History
      */
     protected $_statusHistory;

     /**
      * Grid Pool
      *
      * @var GridPool
      */
     protected $_gridPool;

     /**
      * Data Processor
      *
      * @var DataProcessor
      */
     protected $_dataProcessor;

     /**
      * @var TimezoneInterface
      */
     private $localeDate;

     /**
      * Order constructor.
      * @param Context $context
      * @param EntityFactory $entityFactory
      * @param ItemFactory $itemFactory
      * @param AddressFactory $addressFactory
      * @param ShipmentFactory $shipmentFactory
      * @param ShipmentItemFactory $shipmentItemFactory
      * @param ShipmentTrackFactory $shipmentTrackFactory
      * @param ShipmentCommentFactory $shipmentCommentFactory
      * @param PaymentFactory $paymentFactory
      * @param TransactionFactory $transactionFactory
      * @param InvoiceFactory $invoiceFactory
      * @param InvoiceItemFactory $invoiceItemFactory
      * @param InvoiceCommentFactory $invoiceCommentFactory
      * @param CreditmemoFactory $creditmemoFactory
      * @param CreditmemoItemFactory $creditmemoItemFactory
      * @param CreditmemoCommentFactory $creditmemoCommentFactory
      * @param CreditmemoGeneratorFactory $creditmemoGeneratorFactory
      * @param TaxFactory $taxFactory
```

```
     * @param TaxItemFactory $taxItemFactory
     * @param StatusHistoryFactory $statusHistory
     * @param GridPool $gridPool
     * @param DataProcessor $dataProcessor
     * @param Helper $helper
     * @param TimezoneInterface $localeDate
     * @throws LocalizedException
     */
    public function __construct(
        Context $context,
        EntityFactory $entityFactory,
        ItemFactory $itemFactory,
        AddressFactory $addressFactory,
        ShipmentFactory $shipmentFactory,
        ShipmentItemFactory $shipmentItemFactory,
        ShipmentTrackFactory $shipmentTrackFactory,
        ShipmentCommentFactory $shipmentCommentFactory,
        PaymentFactory $paymentFactory,
        TransactionFactory $transactionFactory,
        InvoiceFactory $invoiceFactory,
        InvoiceItemFactory $invoiceItemFactory,
        InvoiceCommentFactory $invoiceCommentFactory,
        CreditmemoFactory $creditmemoFactory,
        CreditmemoItemFactory $creditmemoItemFactory,
        CreditmemoCommentFactory $creditmemoCommentFactory,
        CreditmemoGeneratorFactory $creditmemoGeneratorFactory,
        TaxFactory $taxFactory,
        TaxItemFactory $taxItemFactory,
        StatusHistoryFactory $statusHistory,
        GridPool $gridPool,
        DataProcessor $dataProcessor,
        \Ivan\ImportExport\Model\ResourceModel\Order\Helper
$resourceHelper,
        Helper $helper,
        TimezoneInterface $localeDate
    ) {
        $this->_logger = $context->getLogger();
        $this->_order = $entityFactory->create();
        $this->_item = $itemFactory->create();
        $this->_address = $addressFactory->create();
        $this->_shipment = $shipmentFactory->create();
        $this->_shipmentItem = $shipmentItemFactory->create();
        $this->_shipmentTrack = $shipmentTrackFactory->create();
        $this->_shipmentComment = $shipmentCommentFactory->create();
        $this->_payment = $paymentFactory->create();
        $this->_transaction = $transactionFactory->create();
        $this->_invoice = $invoiceFactory->create();
        $this->_invoiceItem = $invoiceItemFactory->create();
        $this->_invoiceComment = $invoiceCommentFactory->create();
        $this->_creditmemo = $creditmemoFactory->create();
        $this->_creditmemoItem = $creditmemoItemFactory->create();
        $this->_creditmemoComment = $creditmemoCommentFactory-
>create();
        $this->_creditmemoGenerator = $creditmemoGeneratorFactory-
>create();
        $this->_tax = $taxFactory->create();
        $this->_taxItem = $taxItemFactory->create();
        $this->_statusHistory = $statusHistory->create();
        $this->_gridPool = $gridPool;
        $this->_dataProcessor = $dataProcessor;
        $this->_helper = $helper;
        $this->output = $context->getOutput();

        parent::__construct(
            $context->getJsonHelper(),
            $context->getImportExportData(),
            $context->getDataSourceModel(),
            $context->getConfig(),
            $context->getResource(),
            $resourceHelper,
            $context->getStringUtils(),
            $context->getErrorAggregator()
        );
        $this->_connection = $context->getResource()-
>getConnection('sales');
        $this->localeDate = $localeDate;
    }

    /**
     * Retrieve All Fields Source
     *
     * @return array
     */
    public function getAllFields()
    {
        $fields = [];
        foreach ($this->getChildren() as $adapter) {
            $fields = array_merge($fields, $adapter->getAllFields());
        }
        return $fields;
    }

    /**
     * Retrieve Children Adapters
     *
     * @return array
     */
    public function getChildren()
    {
        return [
            $this->_order,
            $this->_item,
            $this->_address,
            $this->_shipment,
            $this->_shipmentItem,
            $this->_shipmentTrack,
            $this->_shipmentComment,
            $this->_payment,
            $this->_transaction,
            $this->_invoice,
            $this->_invoiceItem,
            $this->_invoiceComment,
            $this->_creditmemo,
            $this->_creditmemoItem,
            $this->_creditmemoComment,
            $this->_creditmemoGenerator,
            $this->_tax,
            $this->_taxItem,
            $this->_statusHistory
        ];
    }

    /**
     * Import Data Rows
     *
     * @return boolean
     */
    protected function _importData()
    {
        $this->_dataProcessor->setFileName(
            $this->_dataSourceModel->getFile()
        );
        /* import data */
        $this->_connection->beginTransaction();
        try {
            if ($this->_order->importData()) {
                list($orderIds, $orderItemIds) = $this-
>_importOrderItem();
                $this->_importAddress($orderIds);
                $this->_importShipment($orderIds, $orderItemIds);
                $this->_importPayment($orderIds);
                $this->_importInvoice($orderIds, $orderItemIds);
                $this->_importCreditmemo($orderIds, $orderItemIds);
                $this->_importTax($orderIds, $orderItemIds);
                $this->_importStatusHistory($orderIds);
                /* refresh grid and grid archive(ee) */
                foreach ($orderIds as $incrementId => $orderId) {
                    $this->_gridPool->refreshByOrderId($orderId);
                    $this->addLogWriteln(__('order with order id:
%1', $incrementId), $this->getOutput(), 'info');
                }
            }
            $this->_connection->commit();
        } catch (Exception $e) {
            $this->addLogWriteln(__('Sorry, but the data is
invalid'), $this->output, 'error');
            $this->_connection->rollBack();
            return false;
        }
        return true;
    }

    /**
     * Import Order Item Data
     *
     * @return array
     */
    protected function _importOrderItem()
    {
        $orderIds = $this->_dataProcessor->merge(
            $this->_order->getOrderIdsMap(),
            'orderIds'
        );
        /* order item */
        $this->_item->setOrderIdsMap($orderIds)
            ->importData();

        $orderItemIds = $this->_dataProcessor->merge(
            $this->_item->getItemIdsMap(),
            'orderItemIds'
        );
        return [$orderIds, $orderItemIds];
    }

    /**
     * Import Address Data
     *
     * @param array $orderIds
     * @return void
     */
    protected function _importAddress(array $orderIds)
    {
        $this->_address->setOrderIdsMap($orderIds)
            ->importData();

        $this->createShippingAddresses($orderIds);
        $this->createAddresses($orderIds);
    }

    /**
     * Create shipping addresses
     *
     * @param $orderIds
     * @return void
     */
    protected function createShippingAddresses($orderIds)
    {
        $orderTable = $this->_address->getOrderTable();
        $addressTable = $this->_address->getMainTable();
        $select = $this->getConnection()
```

```
            ->select()
            ->from(
                ['o' => $orderTable],
                []
            )
            ->joinLeft(
                ['b' => $addressTable],
                'b.parent_id=o.entity_id AND
b.address_type="billing"',
                ['b.*']
            )
            ->joinLeft(
                ['s' => $addressTable],
                's.parent_id=o.entity_id AND
s.address_type="shipping"',
                []
            )
            ->where('s.entity_id IS NULL')
            ->where('o.is_virtual = ?', 0)
            ->where('o.entity_id IN (?)', $orderIds);

        try {
            $addresses = $this->getConnection()->fetchAll($select);
            foreach ($addresses as $address) {
                $address['entity_id'] = null;
                $address['address_type'] = 'shipping';
                $this->getConnection()->insert(
                    $addressTable,
                    $address
                );

                $addressId = $this->getConnection()-
>lastInsertId($addressTable);
                $this->getConnection()->update(
                    $orderTable,
                    ['shipping_address_id' => $addressId],
                    ['entity_id = ?' => $address['parent_id']]
                );
            }
        } catch (\Exception $e) {
            $this->_logger->error($e->getMessage());
        }
    }

    /**
     * Create default orders addresses for orders without any
address.
     * Order addresses are required entities.
     *
     * @param $orderIds
     *
     * @return array
     */
    protected function createAddresses($orderIds)
    {
        $ordersWoAddresses = [];
        $resultIds = [];
        $addressData = [];
        $orderTable = $this->_address->getOrderTable();
        $addressTable = $this->_address->getMainTable();

        $select = $this->getConnection()
            ->select()
            ->from(
                ['so' => $orderTable],
                ['entity_id', 'customer_id', 'customer_email',
'customer_firstname', 'customer_lastname']
            )
            ->joinLeft(
                ['soa' => $addressTable],
                'so.entity_id  = soa.parent_id',
                []
            )
            ->where('soa.entity_id IS NULL')
            ->where('so.entity_id IN (?)', $orderIds);

        try {
            $ordersWoAddresses = $this->getConnection()-
>fetchAll($select);
        } catch (Exception $e) {
            $this->_logger->error($e->getMessage());
        }

        foreach ($ordersWoAddresses as $row) {
            /**
             * @see \Magento\Sales\Model\Order\Payment::place() for
required columns
             */
            $addressData[] = [
                'parent_id' => $row['entity_id'],
                'customer_address_id' => 0,
                'customer_id' => $row['customer_id'],
                'email' => $row['customer_email'],
                'firstname' => $row['customer_firstname'],
                'lastname' => $row['customer_lastname'],
                'address_type' => 'shipping'
            ];
            $addressData[] = [
                'parent_id' => $row['entity_id'],
                'customer_address_id' => 0,
                'customer_id' => $row['customer_id'],
                'email' => $row['customer_email'],
                'firstname' => $row['customer_firstname'],
                'lastname' => $row['customer_lastname'],
                'address_type' => 'billing'
            ];
        }

        if ($addressData) {
            foreach ($addressData as $data) {
                $this->getConnection()->insert(
                    $addressTable,
                    $data
                );
                $resultIds[] = $this->getConnection()-
>lastInsertId($addressTable);
            }
        }

        return $resultIds;
    }

    /**
     * Import Shipment Data
     *
     * @param array $orderIds
     * @param array $orderItemIds
     * @return void
     */
    protected function _importShipment(array $orderIds, array
$orderItemIds)
    {
        $this->_shipment->setOrderIdsMap($orderIds)->importData();
        $shipmentIds = $this->_dataProcessor->merge(
            $this->_shipment->getShipmentIdsMap(),
            'shipmentIds'
        );
        /* shipment item */
        $this->_shipmentItem
            ->setShipmentIdsMap($shipmentIds)
            ->setItemIdsMap($orderItemIds)
            ->importData();
        /* shipment track */
        $this->_shipmentTrack
            ->setShipmentIdsMap($shipmentIds)
            ->setOrderIdsMap($orderIds)
            ->importData();
        /* shipment comment */
        $this->_shipmentComment
            ->setShipmentIdsMap($shipmentIds)
            ->importData();
    }

    /**
     * Import Payment Data
     *
     * @param array $orderIds
     * @return void
     */
    protected function _importPayment(array $orderIds)
    {
        $this->_payment->setOrderIdsMap($orderIds)->importData();

        $paymentIds = $this->_payment->getPaymentIdsMap();
        $newPaymentIds = $this->createPayments($orderIds);
        $paymentIds += $newPaymentIds;

        $this->_transaction
            ->setOrderIdsMap($orderIds)
            ->setPaymentIdsMap($paymentIds)
            ->setTransactionIdsMap([])
            ->importData();
    }

    /**
     * Create default payments for orders without any payment.
     * Order payments are required entities.
     *
     * @param $orderIds
     *
     * @return array
     */
    protected function createPayments($orderIds)
    {
        $ordersWoPayments = [];
        $resultIds = [];
        $paymentData = [];
        $orderTable = $this->_payment->getOrderTable();
        $paymentTable = $this->_payment->getMainTable();

        $select = $this->getConnection()
            ->select()
            ->from(
                ['so' => $orderTable],
                ['entity_id', 'total_due', 'base_total_due',
'shipping_amount', 'base_shipping_amount']
            )
            ->joinLeft(
                ['sop' => $paymentTable],
                'so.entity_id  = sop.parent_id',
                []
            )
            ->where('sop.entity_id IS NULL')
            ->where('so.entity_id IN (?)', $orderIds);

        try {
            $ordersWoPayments = $this->getConnection()-
>fetchAll($select);
        } catch (Exception $e) {
            $this->_logger->error($e->getMessage());
```

```
        }

        foreach ($ordersWoPayments as $row) {
            /**
             * @see \Magento\Sales\Model\Order\Payment::place() for
required columns
             */
            $paymentData[] = [
                'parent_id' => $row['entity_id'],
                'amount_ordered' => $row['total_due'],
                'base_amount_ordered' => $row['base_total_due'],
                'shipping_amount' => $row['shipping_amount'],
                'base_shipping_amount' =>
$row['base_shipping_amount'],
                'method' => Checkmo::PAYMENT_METHOD_CHECKMO_CODE
            ];
        }

        if ($paymentData) {
            foreach ($paymentData as $data) {
                $this->getConnection()->insert(
                    $paymentTable,
                    $data
                );
                $resultIds[] = $this->getConnection()-
>lastInsertId($paymentTable);
            }
        }

        return $resultIds;
    }

    protected function getConnection()
    {
        return $this->_connection;
    }

    /**
     * Import Invoice Data
     *
     * @param array $orderIds
     * @param array $orderItemIds
     * @return void
     */
    protected function _importInvoice(array $orderIds, array
$orderItemIds)
    {
        $this->_invoice->setOrderIdsMap($orderIds)->importData();
        $invoiceIds = $this->_dataProcessor->merge(
            $this->_invoice->getInvoiceIdsMap(),
            'invoiceIds'
        );
        /* invoice item */
        $this->_invoiceItem
            ->setInvoiceIdsMap($invoiceIds)
            ->setItemIdsMap($orderItemIds)
            ->importData();
        /* invoice comment */
        $this->_invoiceComment
            ->setInvoiceIdsMap($invoiceIds)
            ->importData();
    }

    /**
     * Import Creditmemo Data
     *
     * @param array $orderIds
     * @param array $orderItemIds
     * @return void
     */
    protected function _importCreditmemo(array $orderIds, array
$orderItemIds)
    {
        $this->_creditmemo->setOrderIdsMap($orderIds)->importData();
        $creditmemoIds = $this->_dataProcessor->merge(
            $this->_creditmemo->getCreditmemoIdsMap(),
            'creditmemoIds'
        );
        /* creditmemo item */
        $this->_creditmemoItem
            ->setCreditmemoIdsMap($creditmemoIds)
            ->setItemIdsMap($orderItemIds)
            ->importData();
        /* creditmemo comment */
        $this->_creditmemoComment
            ->setCreditmemoIdsMap($creditmemoIds)
            ->importData();
        /* creditmemo generator */
        $this->_creditmemoGenerator->importData();
    }

    /**
     * Import Tax Data
     *
     * @param array $orderIds
     * @param array $orderItemIds
     * @return void
     */
    protected function _importTax(array $orderIds, array
$orderItemIds)
    {
        $this->_tax->setOrderIdsMap($orderIds)->importData();
        $taxIds = $this->_dataProcessor->merge(
            $this->_tax->getTaxIdsMap(),
            'taxIds'
        );
        /* tax item */
        $this->_taxItem
            ->setTaxIdsMap($taxIds)
            ->setItemIdsMap($orderItemIds)
            ->importData();
    }

    /**
     * Import Status History Data
     *
     * @param array $orderIds
     * @return void
     */
    protected function _importStatusHistory(array $orderIds)
    {
        $this->_statusHistory->setOrderIdsMap($orderIds)-
>importData();
    }

    /**
     * Validate Data Row
     *
     * @param array $rowData
     * @param int $rowNumber
     * @return boolean
     */
    public function validateRow(array $rowData, $rowNumber)
    {
        foreach ($this->getChildren() as $adapter) {
            $tempData = $adapter->prepareRowData($rowData);

            /**
             * Check if array is not empty with array_filter
function.
             */
            if ($tempData && array_filter($tempData) && !$adapter-
>validateRow($tempData, $rowNumber)) {
                return false;
            }
        }
        return true;
    }

    /**
     * Retrieve Entity Type Code
     *
     * @return string
     */
    public function getEntityTypeCode()
    {
        return self::ENTITY_TYPE_CODE;
    }

    /**
     * Save Validated Bunches
     *
     * @return $this
     * @throws LocalizedException
     */
    protected function _saveValidatedBunches()
    {
        $source = $this->_getSource();
        $currentDataSize = 0;
        $bunchRows = [];
        $startNewBunch = false;
        $nextRowBackup = [];
        $maxDataSize = $this->_resourceHelper->getMaxDataSize();
        $bunchSize = $this->_importExportData->getBunchSize();

        $source->rewind();
        $this->_dataSourceModel->cleanBunches();
        $file = null;
        $jobId = null;
        if (isset($this->_parameters['file'])) {
            $file = $this->_parameters['file'];
        }
        if (isset($this->_parameters['job_id'])) {
            $jobId = $this->_parameters['job_id'];
        }

        while ($source->valid() || $bunchRows) {
            if ($startNewBunch || !$source->valid()) {
                $this->_dataSourceModel->saveBunches(
                    $this->getEntityTypeCode(),
                    $this->getBehavior(),
                    $jobId,
                    $file,
                    $bunchRows
                );
                $bunchRows = $nextRowBackup;
                $currentDataSize = strlen($this->jsonHelper-
>jsonEncode($bunchRows));
                $startNewBunch = false;
                $nextRowBackup = [];
            }

            if ($source->valid()) {
                try {
                    $rowData = $source->current();
                } catch (InvalidArgumentException $e) {
                    $this->addRowError($e->getMessage(), $this-
>_processedRowsCount);
                    $this->_processedRowsCount++;
                    $source->next();
```

```
                    continue;
                }
                $rowData = $this->customBunchesData($rowData);
                $this->_processedRowsCount++;
                if ($this->validateRow($rowData, $source->key())) {
                    $rowSize = strlen($this->jsonHelper-
>jsonEncode($rowData));

                    $isBunchSizeExceeded = $bunchSize > 0 &&
count($bunchRows) >= $bunchSize;

                    if (!empty($rowData['increment_id']) &&
                        ($currentDataSize + $rowSize >= $maxDataSize
|| $isBunchSizeExceeded)
                    ) {
                        $startNewBunch = true;
                        $nextRowBackup = [$source->key() =>
$rowData];
                    } else {
                        $bunchRows[$source->key()] = $rowData;
                        $currentDataSize += $rowSize;
                    }
                }
                $source->next();
            }
        }
        return $this;
    }

    /**
     * Output Model Setter
     *
     * @param $output
     * @return $this
     */
    public function setOutput($output)
    {
        $this->output = $output;
        foreach ($this->getChildren() as $adapter) {
            $adapter->setOutput($output);
        }
        return $this;
    }

    /**
     * Logger Model Setter
     *
     * @param $logger
     * @return $this
     */
    public function setLogger($logger)
    {
        $this->_logger = $logger;
        foreach ($this->getChildren() as $adapter) {
            $adapter->setLogger($logger);
        }
        return $this;
    }

    /**
     * Set Data From Outside To Change Behavior
     *
     * @param array $parameters
     * @return $this
     */
    public function setParameters(array $parameters)
    {
        parent::setParameters($parameters);
        foreach ($this->getChildren() as $adapter) {
            $adapter->setParameters($parameters);
        }
        return $this;
    }

    /**
     * Source Model Setter
     *
     * @param AbstractSource $source
     * @return
\Magento\ImportExport\Model\Import\AbstractEntity|AbstractEntity
     */
    public function setSource(AbstractSource $source)
    {
        foreach ($this->getChildren() as $adapter) {
            $adapter->setSource($source);
        }
        return parent::setSource($source);
    }

    /**
     * Error Aggregator Setter
     *
     * @param ProcessingErrorAggregatorInterface $errorAggregator
     * @return $this
     */
    public function setErrorAggregator($errorAggregator)
    {
        $this->errorAggregator = $errorAggregator;
        foreach ($this->getChildren() as $adapter) {
            $adapter->setErrorAggregator($errorAggregator);
            $adapter->initErrorTemplates();
        }
        return $this;
    }

    /**
     * Returns Number Of Checked Entities
     *
     * @return int
     */
    public function getProcessedEntitiesCount()
    {
        $this->_processedEntitiesCount = $this->_order-
>getProcessedEntitiesCount();
        foreach ($this->getChildren() as $adapter) {
            $this->_processedEntitiesCount = max(
                $this->_processedEntitiesCount,
                $adapter->getProcessedEntitiesCount()
            );
        }
        return $this->_processedEntitiesCount;
    }

    /**
     * @param array $rowData
     * @return mixed
     * @throws LocalizedException
     */
    public function customBunchesData(array $rowData)
    {
        $dateFormat = 'Y-m-d H:i:s';
        foreach (['created_at', 'updated_at'] as $key) {
            if (isset($rowData[$key]) && !empty($rowData[$key])) {
                $originalDate = str_replace('/', '-',
$rowData[$key]);
                if (strtotime($originalDate)) {
                    $rowData[$key] = $this->localeDate-
>convertConfigTimeToUtc($originalDate, $dateFormat);
                }
            }
        }
        if (isset($rowData['shipping_method']) &&
!empty($rowData['shipping_method'])) {
            $rowData['shipping_method'] = str_replace(' ', '_',
$rowData['shipping_method']);
        }
        return $rowData;
    }
}
```

```
<?php

namespace Ivan\ImportExport\Model\Import;

use Magento\Framework\Module\Dir\Reader;

/**
 * Class Platforms
 *
 * @package Ivan\ImportExport\Model\Import
 */
class Platforms extends \Magento\Framework\DataObject
{
    const URL_DOWNLOAD = "import/job/download";
    const GITHUB_LINK = "https://github.com/Ivanstudio/magento2-
import-export-sample-files";

    /**
     * @var Reader
     */
    protected $moduleReader;

    /**
     * @var \Magento\Backend\Model\UrlInterface
     */
    protected $backendUrl;

    /**
     * @var array|mixed|null
     */
    protected $platforms;

    /**
     * Platforms constructor.
     * @param Reader $moduleReader
     * @param \Magento\Backend\Model\UrlInterface $backendUrl
     * @param \Ivan\ImportExport\Model\Source\Platform\Config $config
     * @param array $data
     */
    public function __construct(
        \Magento\Framework\Module\Dir\Reader $moduleReader,
        \Magento\Backend\Model\UrlInterface $backendUrl,
        \Ivan\ImportExport\Model\Source\Platform\Config $config,
        array $data = []
    ) {
        parent::__construct($data);
        $this->moduleReader = $moduleReader;
        $this->backendUrl = $backendUrl;
        $this->platforms = $config->get();
    }

    /**
     * @return array
     */
    public function toOptionArrayLinks($entityType =
'catalog_product')
    {
        $list = [];
        $platforms = $this->platforms[$entityType] ?? [];
        foreach ($platforms as $platform => $data) {
```

```
            if (isset($data['links'])) {
                foreach ($data['links'] as $link) {
                    if (isset($link['entity']) && $link['entity'] ===
'github_link') {
                        $list[] = [
                            'label' => __($link['label']),
                            'href' => self::GITHUB_LINK,
                            'type' => $platform,
                            'entity' => isset($link['entity']) ?
$link['entity'] : ''
                        ];
                    } else {
                        $list[] = [
                            'label' => __($link['label']),
                            'href' => $this->backendUrl->getUrl(
                                self::URL_DOWNLOAD,
                                ['type' => $platform .
$link['suffix']]
                            ),
                            'type' => $platform,
                            'entity' => isset($link['entity']) ?
$link['entity'] : ''
                        ];
                    }
                }
            }
        }
        return $list;
    }

    /**
     * @return array
     */
    public function toOptionArrayNames()
    {
        return [];
    }

    /**
     * @return array
     */
    public function toOptionArrayButton()
    {
        $list = [];
        foreach ($this->platforms as $platform => $data) {
            $list[$platform] = [];
            if (isset($data['fields'])) {
                foreach ($data['fields'] as $name => $value) {
                    $list[$platform][] = ['label' =>
$value['reference'], 'value' => $name];
                }
            }
        }

        return $list;
    }

    /**
     * @param $type
     * @return array
     */
    public function getAllData($type)
    {
        $list = [];
        foreach ($this->platforms as $platform => $data) {
            if ($platform == $type) {
                if (isset($data['fields'])) {
                    foreach ($data['fields'] as $name => $value) {
                        $list[$name] = ['reference' =>
$value['reference'],'default' => $value['default']];
                    }
                }
            }
        }

        return $list;
    }

    /**
     * @param $entityType
     * @return array
     */
    public function getPlatformList($entityType)
    {
        $options = [];
        $platforms = $this->platforms[$entityType] ?? [];
        foreach ($platforms as $platform => $data) {
            $options[] = [
                'label' => __($data['label']),
                'value' => $platform
            ];
        }
        return ['options' => $options];
    }
}
```

```
<?php

namespace Ivan\ImportExport\Model\Import;

use
Magento\ImportExport\Model\Import\ErrorProcessing\ProcessingError;

/**
 * Class ProcessingErrorAggregator
 *
 * @package Ivan\ImportExport\Model\Import
 */
class ProcessingErrorAggregator extends
\Magento\ImportExport\Model\Import\ErrorProcessing\ProcessingErrorAggr
egator
{
    /**
     * @param string $errorLevel
     * @param int $count
     * @return ProcessingErrorAggregator
     */
    protected function setProcessErrorStatistics($errorLevel, $count)
    {
        $this->errorStatistics[$errorLevel] = $count;

        return $this;
    }

    public function hasToBeTerminated()
    {
        return $this->isErrorLimitExceeded();
    }

    public function isErrorLimitExceeded()
    {
        $isExceeded = false;
        $errorsCount = $this-
>getErrorsCount([ProcessingError::ERROR_LEVEL_CRITICAL]);
        if ($errorsCount > 0
            && $this->validationStrategy ==
self::VALIDATION_STRATEGY_STOP_ON_ERROR
            && $errorsCount > $this->allowedErrorsCount
        ) {
            $isExceeded = true;
        }

        return $isExceeded;
    }

    protected function isErrorAlreadyAdded($rowNum, $errorCode,
$columnName = null)
    {
        return false;
    }
}
```

```
<?php
declare(strict_types=1);

namespace Ivan\ImportExport\Model\Import;

use function array_diff;
use function array_keys;
use function array_map;
use function array_merge;
use function array_values;
use Exception;
use function explode;
use Ivan\ImportExport\Api\UrlKeyManagerInterface;
use Ivan\ImportExport\Helper\Additional;
use Ivan\ImportExport\Helper\Data as IvanImportExportData;
use Ivan\ImportExport\Model\Cache\Type\ImportProduct as
ImportProductCache;
use Ivan\ImportExport\Model\Export\RowCustomizer\ProductVideo;
use Ivan\ImportExport\Model\Import;
use Ivan\ImportExport\Model\Import\Product\CategoryProcessor;
use Ivan\ImportExport\Model\Import\Product\Image as ImportImage;
use Ivan\ImportExport\Model\Import\Product\ImageProcessor as
ImportImageProcessor;
use
Ivan\ImportExport\Model\Import\Product\Integration\IntegrationInterfac
e;
use
Ivan\ImportExport\Model\Import\Product\Integration\MageArrayMarketplac
e;
use
Ivan\ImportExport\Model\Import\Product\Integration\WebkulMarketplace;
use Ivan\ImportExport\Model\Import\Product\MediaVideoGallery;
use Ivan\ImportExport\Model\Import\Product\OptionFactory;
use
Ivan\ImportExport\Model\Import\Product\Price\Rule\ConditionFactory as
ConditionFactoryAlias;
use Ivan\ImportExport\Model\Import\Product\Type\Downloadable;
use Ivan\ImportExport\Model\Job;
use Ivan\ImportExport\Model\ResourceModel\Job\CollectionFactory;
use Ivan\ImportExport\Model\Source\Import\Config;
use Ivan\ImportExport\Model\Source\Type\AbstractType;
use Ivan\ImportExport\Model\Translation\Translator;
use Ivan\ImportExport\Traits\Import\Entity as ImportTrait;
use
Ivan\ImportExport\Ui\Component\Listing\Column\Entity\Import\Attributes
\SystemOptions;
use
Ivan\ImportExport\Ui\Component\Listing\Column\Import\Source\Configurab
le\Type\Options as TypeOptions;
use function implode;
use function in_array;
use function interface_exists;
use InvalidArgumentException;
use function is_array;
use function is_int;
use Magento\Bundle\Model\Product\Price as BundlePrice;
use Magento\BundleImportExport\Model\Import\Product\Type\Bundle;
use Magento\Catalog\Api\Data\CategoryLinkInterface;
```

```
 use Magento\Catalog\Api\Data\ProductInterface;
 use Magento\Catalog\Api\ProductRepositoryInterface;
 use Magento\Catalog\Helper\Data as CatalogHelperData;
 use Magento\Catalog\Helper\Product as CatalogHelperProduct;
 use Magento\Catalog\Model\CategoryLinkRepository;
 use Magento\Catalog\Model\Config as CatalogConfig;
 use Magento\Catalog\Model\Product\Action;
 use Magento\Catalog\Model\Product\ActionFactory;
 use Magento\Catalog\Model\Product\Attribute\Backend\Sku;
 use Magento\Catalog\Model\Product\Attribute\Source\Status;
 use Magento\Catalog\Model\Product\Link as ProductLink;
 use Magento\Catalog\Model\Product\Media\ConfigInterface;
 use Magento\Catalog\Model\Product\Url;
 use Magento\Catalog\Model\Product\Visibility;
 use Magento\Catalog\Model\ResourceModel\Eav\AttributeFactory;
 use Magento\Catalog\Model\ResourceModel\Product\Collection;
 use Magento\Catalog\Model\ResourceModel\Product\CollectionFactory as
ProductCollectionFactory;
 use Magento\Catalog\Model\ResourceModel\Product\LinkFactory;
 use Magento\CatalogImportExport\Model\Import\Product as
MagentoProduct;
 use
Magento\CatalogImportExport\Model\Import\Product\ImageTypeProcessor;
 use
Magento\CatalogImportExport\Model\Import\Product\MediaGalleryProcessor
;
 use
Magento\CatalogImportExport\Model\Import\Product\RowValidatorInterface
as ValidatorInterface;
 use Magento\CatalogImportExport\Model\Import\Product\SkuProcessor;
 use Magento\CatalogImportExport\Model\Import\Product\StoreResolver;
 use
Magento\CatalogImportExport\Model\Import\Product\TaxClassProcessor;
 use Magento\CatalogImportExport\Model\Import\Product\Type\Factory;
 use Magento\CatalogImportExport\Model\Import\Product\Validator;
 use
Magento\CatalogImportExport\Model\Import\Proxy\Product\ResourceModel;
 use
Magento\CatalogImportExport\Model\Import\Proxy\Product\ResourceModelFa
ctory;
 use Magento\CatalogImportExport\Model\Import\Proxy\ProductFactory;
 use Magento\CatalogImportExport\Model\StockItemImporterInterface;
 use Magento\CatalogInventory\Api\StockConfigurationInterface;
 use Magento\CatalogInventory\Api\StockRegistryInterface;
 use Magento\CatalogInventory\Model\ResourceModel\Stock\Item;
 use Magento\CatalogInventory\Model\ResourceModel\Stock\ItemFactory;
 use Magento\CatalogInventory\Model\Spi\StockStateProviderInterface;
 use Magento\Customer\Model\GroupFactory;
 use Magento\Eav\Model\Entity\Attribute\AbstractAttribute;
 use Magento\Eav\Model\EntityFactory;
 use
Magento\Eav\Model\ResourceModel\Entity\Attribute\Group\CollectionFacto
ry as AttributeGroupCollectionFactory;
 use
Magento\Eav\Model\ResourceModel\Entity\Attribute\Set\CollectionFactory
as AttributeSetCollectionFactory;
 use Magento\Framework\App\CacheInterface;
 use Magento\Framework\App\Config\ScopeConfigInterface;
 use Magento\Framework\App\Filesystem\DirectoryList;
 use Magento\Framework\App\ObjectManager;
 use Magento\Framework\App\ProductMetadataInterface;
 use Magento\Framework\DB\Adapter\AdapterInterface;
 use Magento\Framework\DB\Adapter\ConnectionException;
 use Magento\Framework\DB\Adapter\DeadlockException;
 use Magento\Framework\DB\Adapter\LockWaitException;
 use Magento\Framework\Event\ManagerInterface;
 use Magento\Framework\Exception\LocalizedException;
 use Magento\Framework\Exception\NoSuchEntityException;
 use Magento\Framework\Filesystem;
 use Magento\Framework\Indexer\IndexerRegistry;
 use Magento\Framework\Intl\DateTimeFactory;
 use Magento\Framework\MessageQueue\PublisherInterface;
 use Magento\Framework\Model\ResourceModel\Db\ObjectRelationProcessor;
 use
Magento\Framework\Model\ResourceModel\Db\TransactionManagerInterface;
 use Magento\Framework\Module\Manager;
 use Magento\Framework\Stdlib\DateTime;
 use Magento\Framework\Stdlib\DateTime\TimezoneInterface;
 use Magento\ImportExport\Model\Import\Config as ImportConfig;
 use Magento\ImportExport\Model\Import\Entity\AbstractEntity;
 use
Magento\ImportExport\Model\Import\ErrorProcessing\ProcessingError;
 use
Magento\ImportExport\Model\Import\ErrorProcessing\ProcessingErrorAggre
gatorInterface;
 use Magento\Store\Model\ScopeInterface;
 use Magento\Store\Model\Store;
 use Magento\Store\Model\StoreManagerInterface;
 use Magento\Store\Model\WebsiteFactory;
 use Magento\Swatches\Helper\Data;
 use Magento\Swatches\Helper\Media;
 use Magento\Swatches\Model\ResourceModel\Swatch\CollectionFactory as
SwatchCollectionFactory;
 use Magento\Swatches\Model\Swatch;
 use Magento\Tax\Model\ClassModel;
 use Magento\Tax\Model\ResourceModel\TaxClass\CollectionFactory as
TaxClassCollectionFactory;
 use function mb_strtolower;
 use function strlen;
 use function strtolower;
 use function substr;
 use Symfony\Component\Console\Output\ConsoleOutput;
 use function version_compare;
 use Zend\Serializer\Serializer;
 use Zend_Db_Select;
use Zend_Validate_Exception;
use Zend_Validate_Regex;

/**
 * Import entity product model
 *
 * @api
 *
 * @SuppressWarnings(PHPMD.TooManyFields)
 * @SuppressWarnings(PHPMD.ExcessiveClassComplexity)
 * @SuppressWarnings(PHPMD.CouplingBetweenObjects)
 * @since 100.0.2
 */
class Product extends MagentoProduct
{
    use ImportTrait;

    /**
     * Default website id
     */
    const DEFAULT_WEBSITE_ID = 1;
    /**
     * Used when create new attributes in column name
     */
    const ATTRIBUTE_SET_GROUP = 'attribute_set_group';
    /**
     * Attribute sets column name
     */
    const ATTRIBUTE_SET_COLUMN = 'attribute_set';

    /**
     * Maximum number of database retries
     */
    const MAX_DB_RETRIES = 5;

    protected $mediaProcessor;

    protected $classTaxNames = [];

    /**
     * @var ProductMetadataInterface
     */
    public $productMetadata;
    public $onlyUpdate = 0;
    protected $onlyAdd = false;
    public static $addFields = [
        'manage_stock',
        'use_config_manage_stock',
        'qty',
        'min_qty',
        'use_config_min_qty',
        'min_sale_qty',
        'use_config_min_sale_qty',
        'max_sale_qty',
        'use_config_max_sale_qty',
        'is_qty_decimal',
        'backorders',
        'use_config_backorders',
        'notify_stock_qty',
        'use_config_notify_stock_qty',
        'enable_qty_increments',
        'use_config_enable_qty_inc',
        'qty_increments',
        'use_config_qty_increments',
        'is_in_stock',
        'low_stock_date',
        'stock_status_changed_auto',
        'is_decimal_divided',
        'has_options',
        'tax_class_name',
        self::COL_STORE_VIEW_CODE,
        'attribute_set_code',
        'configurable_variations',
        'configurable_variation_labels',
        'associated_skus',
        'base_image_label',
        'additional_images',
        'additional_image_labels',
        'small_image_label',
        'thumbnail_image_label',
        'swatch_image',
        'swatch_image_label',
        'remove_images',
        ProductVideo::VIDEO_URL_COLUMN,
        WebkulMarketplace::VENDOR_ID,
        WebkulMarketplace::COL_UNASSIGN_SELLER,
        MageArrayMarketplace::VENDOR_ID,
        MageArrayMarketplace::MAGE_PRICE_COMPARE,
        'additional_attributes',
        'custom_options'
    ];

    public static $specialAttributes = [
        self::COL_STORE,
        self::COL_ATTR_SET,
        self::COL_TYPE,
        self::COL_CATEGORY,
        self::COL_CATEGORY . '_position',
        'product_websites',
        self::COL_PRODUCT_WEBSITES,
        '_tier_price_website',
        '_tier_price_customer_group',
        '_tier_price_qty',
        '_tier_price_price',
        '_related_sku',
```

```
    '_related_position',
    '_crosssell_sku',
    '_crosssell_position',
    '_upsell_sku',
    '_upsell_position',
    '_custom_option_store',
    '_custom_option_type',
    '_custom_option_title',
    '_custom_option_is_required',
    '_custom_option_price',
    '_custom_option_sku',
    '_custom_option_max_characters',
    '_custom_option_sort_order',
    '_custom_option_file_extension',
    '_custom_option_image_size_x',
    '_custom_option_image_size_y',
    '_custom_option_row_title',
    '_custom_option_row_price',
    '_custom_option_row_sku',
    '_custom_option_row_sort',
    '_media_attribute_id',
    self::COL_MEDIA_IMAGE,
    '_media_label',
    '_media_position',
    '_media_is_disabled',
    '_tier_price_value_type',
    'product_online',
    'update_attribute_set',
];

/**
 * @var UploaderFactory
 */
protected $_uploaderFactory;

/**
 * @var Product\CategoryProcessor
 */
protected $categoryProcessor;

/**
 * @var IvanImportExportData
 */
protected $helper;
/**
 * @var Additional
 */
protected $additional;
/**
 * @var AbstractType
 */
protected $sourceType;
/**
 * @var AttributeFactory
 */
protected $attributeFactory;
/**
 * @var EntityFactory
 */
protected $eavEntityFactory;
/**
 * @var AttributeGroupCollectionFactory
 */
protected $groupCollectionFactory;
/**
 * @var array
 */
protected $_attributeSetGroupCache;
/**
 * @var CatalogHelperProduct
 */
protected $productHelper;

protected $_debugMode;
/**
 * @var Config
 */
protected $fireImportConfig;
/**
 * @var ProductRepositoryInterface
 */
protected $productRepository;

/**
 * @var ProductCollectionFactory
 */
protected $collectionFactory;

/**
 * @var Data
 */
protected $swatchesHelperData;

/**
 * Helper to move image from tmp to catalog
 *
 * @var Media
 */
protected $swatchHelperMedia;

/**
 * @var SwatchCollectionFactory
 */
protected $swatchCollectionFactory;

/**
 * @var ConfigInterface
 */
protected $mediaConfig;
/**
 * @var GroupFactory
 */
protected $groupFactory;
/**
 * @var WebsiteFactory
 */
protected $websiteFactory;

/**
 * @var TaxClassCollectionFactory
 */
protected $collectionTaxFactory;
/**
 * @var StoreManagerInterface
 */
protected $storeManager;
protected $notValidedSku = [];
protected $productLinkData = [];
protected $productLink;
protected $priceRuleConditionFactory;
protected $platform;
/** @var Manager */
protected $manager;
/**
 * @var UrlKeyManagerInterface
 */
protected $urlKeyManager;
/**
 * @var ActionFactory
 */
protected $productActionFactory;

private $cachedSwatchOptions = [];
private $importCollection;
private $_isRowCategoryMapped;
private $lastSku;

/**
 * Product entity identifier field
 *
 * @var string
 */
private $productEntityIdentifierField;

/**
 * Product entity link field
 *
 * @var string
 */
private $productEntityLinkField;

/** @var Translator */
protected $translator;

/**
 *
 * @var CategoryLinkRepository
 */
protected $categoryLinkRepository;

/**
 * Stock Item Importer
 *
 * @var StockItemImporterInterface
 */
private $stockItemImporter;

protected $categoryProductPosition = [];

/**
 * @var ProductCollectionFactory
 */
private $productCollectionFactory;

/** @var array */
protected $integrations;

/**
 * @var PublisherInterface
 */
protected $publisher;

/**
 * @var ImportImage
 */
protected $importImage;

/**
 * @var ImportImageProcessor
 */
protected $importImageProcessor;

/**
 * @var CacheInterface
 */
protected $cache;

/**
 * @var array
 */
```

```
    protected $originalImportRows = [];

    /**
     * @var array
     */
    protected $getProductIdByRowId = [];

    /**
     * @var array
     */
    protected $adminAttributeValue = [];

    /**
     * Product constructor.
     * @param Context $context
     * @param IvanImportExportData $helper
     * @param Additional $additional
     * @param ManagerInterface $eventManager
     * @param StockRegistryInterface $stockRegistry
     * @param StockConfigurationInterface $stockConfiguration
     * @param StockStateProviderInterface $stockStateProvider
     * @param CatalogHelperData $catalogData
     * @param ImportConfig $importConfig
     * @param Config $fireImportConfig
     * @param ResourceModelFactory $resourceFactory
     * @param OptionFactory $optionFactory
     * @param AttributeSetCollectionFactory $setColFactory
     * @param Factory $productTypeFactory
     * @param LinkFactory $linkFactory
     * @param ProductFactory $proxyProdFactory
     * @param Filesystem $filesystem
     * @param ItemFactory $stockResItemFac
     * @param TimezoneInterface $localeDate
     * @param DateTime $dateTime
     * @param IndexerRegistry $indexerRegistry
     * @param StoreResolver $storeResolver
     * @param SkuProcessor $skuProcessor
     * @param Validator $validator
     * @param ObjectRelationProcessor $objectRelationProcessor
     * @param TransactionManagerInterface $transactionManager
     * @param TaxClassProcessor $taxClassProcessor
     * @param ScopeConfigInterface $scopeConfig
     * @param Url $productUrl
     * @param AttributeFactory $attributeFactory
     * @param EntityFactory $eavEntityFactory
     * @param AttributeGroupCollectionFactory $groupCollectionFactory
     * @param CatalogHelperProduct $productHelper
     * @param ProductMetadataInterface $productMetadata
     * @param ProductRepositoryInterface $productRepository
     * @param ProductCollectionFactory $collectionFactory
     * @param GroupFactory $groupFactory
     * @param WebsiteFactory $websiteFactory
     * @param CategoryProcessor $categoryProcessor
     * @param UploaderFactory $uploaderFactory
     * @param TaxClassCollectionFactory $collectionTaxFactory
     * @param StoreManagerInterface $storeManager
     * @param CollectionFactory $importCollectionFactory
     * @param ConditionFactoryAlias $priceRuleConditionFactory
     * @param Data $swatchesHelperData
     * @param Media $swatchHelperMedia
     * @param SwatchCollectionFactory $swatchCollectionFactory
     * @param ConfigInterface $mediaConfig
     * @param Manager $moduleManager
     * @param ProductLink $productLink
     * @param UrlKeyManagerInterface $urlKeyManager
     * @param CategoryLinkRepository $categoryLinkRepository
     * @param ActionFactory $productActionFactory
     * @param Translator $translator
     * @param array $data
     * @param array $integrations
     * @param array $dateAttrCodes
     * @param CatalogConfig|null $catalogConfig
     * @param DateTimeFactory|null $dateTimeFactory
     * @throws LocalizedException
     * @throws Exception
     */
    public function __construct(
        Context $context,
        ImportImage $importImage,
        ImportImageProcessor $importImageProcessor,
        IvanImportExportData $helper,
        CacheInterface $cache,
        Additional $additional,
        ManagerInterface $eventManager,
        StockRegistryInterface $stockRegistry,
        StockConfigurationInterface $stockConfiguration,
        StockStateProviderInterface $stockStateProvider,
        CatalogHelperData $catalogData,
        ImportConfig $importConfig,
        Config $fireImportConfig,
        ResourceModelFactory $resourceFactory,
        Product\OptionFactory $optionFactory,
        AttributeSetCollectionFactory $setColFactory,
        Factory $productTypeFactory,
        LinkFactory $linkFactory,
        ProductFactory $proxyProdFactory,
        Filesystem $filesystem,
        ItemFactory $stockResItemFac,
        TimezoneInterface $localeDate,
        DateTime $dateTime,
        IndexerRegistry $indexerRegistry,
        StoreResolver $storeResolver,
        SkuProcessor $skuProcessor,
        Validator $validator,
        ObjectRelationProcessor $objectRelationProcessor,
        TransactionManagerInterface $transactionManager,
        TaxClassProcessor $taxClassProcessor,
        ScopeConfigInterface $scopeConfig,
        Url $productUrl,
        AttributeFactory $attributeFactory,
        EntityFactory $eavEntityFactory,
        AttributeGroupCollectionFactory $groupCollectionFactory,
        CatalogHelperProduct $productHelper,
        ProductMetadataInterface $productMetadata,
        ProductRepositoryInterface $productRepository,
        ProductCollectionFactory $collectionFactory,
        GroupFactory $groupFactory,
        WebsiteFactory $websiteFactory,
        Product\CategoryProcessor $categoryProcessor,
        UploaderFactory $uploaderFactory,
        TaxClassCollectionFactory $collectionTaxFactory,
        StoreManagerInterface $storeManager,
        CollectionFactory $importCollectionFactory,
        Product\Price\Rule\ConditionFactory
$priceRuleConditionFactory,
        Data $swatchesHelperData,
        Media $swatchHelperMedia,
        SwatchCollectionFactory $swatchCollectionFactory,
        ConfigInterface $mediaConfig,
        Manager $moduleManager,
        ProductLink $productLink,
        UrlKeyManagerInterface $urlKeyManager,
        CategoryLinkRepository $categoryLinkRepository,
        ActionFactory $productActionFactory,
        Translator $translator,
        array $data = [],
        array $integrations = [],
        array $dateAttrCodes = [],
        CatalogConfig $catalogConfig = null,
        DateTimeFactory $dateTimeFactory = null
    ) {
        $this->output = $context->getOutput();
        $this->importImage = $importImage;
        $this->cache = $cache;
        $this->helper = $helper;
        $this->importImageProcessor = $importImageProcessor;
        $this->attributeFactory = $attributeFactory;
        $this->eavEntityFactory = $eavEntityFactory;
        $this->groupCollectionFactory = $groupCollectionFactory;
        $this->productHelper = $productHelper;
        $this->additional = $additional;
        $this->fireImportConfig = $fireImportConfig;
        $this->groupFactory = $groupFactory;
        $this->storeManager = $storeManager;
        $this->priceRuleConditionFactory =
$priceRuleConditionFactory;
        $this->swatchesHelperData = $swatchesHelperData;
        $this->swatchHelperMedia = $swatchHelperMedia;
        $this->swatchCollectionFactory = $swatchCollectionFactory;
        $this->mediaConfig = $mediaConfig;
        $this->translator = $translator;
        $this->productLink = $productLink;

        if (class_exists(MediaGalleryProcessor::class)) {
            $this->mediaProcessor = ObjectManager::getInstance()
                ->get(MediaVideoGallery::class);
            $mediaProcessor = ObjectManager::getInstance()
                ->get(MediaGalleryProcessor::class);
        }

        if (interface_exists(StockItemImporterInterface::class)) {
            $this->stockItemImporter = ObjectManager::getInstance()
                ->get(StockItemImporterInterface::class);
        }
        if (class_exists(ImageTypeProcessor::class)) {
            $imageTypeProcessor = ObjectManager::getInstance()
                ->get(ImageTypeProcessor::class);
        }
        if (interface_exists(PublisherInterface::class)) {
            $this->publisher = ObjectManager::getInstance()
                ->get(PublisherInterface::class);
        }
        if (version_compare($productMetadata->getVersion(), '2.3.0',
'>=')) {
            parent::__construct(
                $context->getJsonHelper(),
                $context->getImportExportData(),
                $context->getDataSourceModel(),
                $context->getConfig(),
                $context->getResource(),
                $context->getResourceHelper(),
                $context->getStringUtils(),
                $context->getErrorAggregator(),
                $eventManager,
                $stockRegistry,
                $stockConfiguration,
                $stockStateProvider,
                $catalogData,
                $importConfig,
                $resourceFactory,
                $optionFactory,
                $setColFactory,
                $productTypeFactory,
                $linkFactory,
                $proxyProdFactory,
                $uploaderFactory,
                $filesystem,
                $stockResItemFac,
                $localeDate,
                $dateTime,
```

```
                $context->getLogger(),
                $indexerRegistry,
                $storeResolver,
                $skuProcessor,
                $categoryProcessor,
                $validator,
                $objectRelationProcessor,
                $transactionManager,
                $taxClassProcessor,
                $scopeConfig,
                $productUrl,
                $data,
                $dateAttrCodes,
                $catalogConfig,
                $imageTypeProcessor,
                $mediaProcessor,
                $this->stockItemImporter,
                $dateTimeFactory,
                $productRepository
            );
        } elseif (version_compare($productMetadata->getVersion(),
'2.2.3', '<=')) {
            parent::__construct(
                $context->getJsonHelper(),
                $context->getImportExportData(),
                $context->getDataSourceModel(),
                $context->getConfig(),
                $context->getResource(),
                $context->getResourceHelper(),
                $context->getStringUtils(),
                $context->getErrorAggregator(),
                $eventManager,
                $stockRegistry,
                $stockConfiguration,
                $stockStateProvider,
                $catalogData,
                $importConfig,
                $resourceFactory,
                $optionFactory,
                $setColFactory,
                $productTypeFactory,
                $linkFactory,
                $proxyProdFactory,
                $uploaderFactory,
                $filesystem,
                $stockResItemFac,
                $localeDate,
                $dateTime,
                $context->getLogger(),
                $indexerRegistry,
                $storeResolver,
                $skuProcessor,
                $categoryProcessor,
                $validator,
                $objectRelationProcessor,
                $transactionManager,
                $taxClassProcessor,
                $scopeConfig,
                $productUrl,
                $data,
                $dateAttrCodes,
                $catalogConfig
            );
        } else {
            parent::__construct(
                $context->getJsonHelper(),
                $context->getImportExportData(),
                $context->getDataSourceModel(),
                $context->getConfig(),
                $context->getResource(),
                $context->getResourceHelper(),
                $context->getStringUtils(),
                $context->getErrorAggregator(),
                $eventManager,
                $stockRegistry,
                $stockConfiguration,
                $stockStateProvider,
                $catalogData,
                $importConfig,
                $resourceFactory,
                $optionFactory,
                $setColFactory,
                $productTypeFactory,
                $linkFactory,
                $proxyProdFactory,
                $uploaderFactory,
                $filesystem,
                $stockResItemFac,
                $localeDate,
                $dateTime,
                $context->getLogger(),
                $indexerRegistry,
                $storeResolver,
                $skuProcessor,
                $categoryProcessor,
                $validator,
                $objectRelationProcessor,
                $transactionManager,
                $taxClassProcessor,
                $scopeConfig,
                $productUrl,
                $data,
                $dateAttrCodes,
                $catalogConfig,
                $mediaProcessor
            );
        }
        $this->setSerializer($context->getSerializer());
        $this->setLogger($context->getLogger());
        $this->_debugMode = $helper->getDebugMode();
        $this->productMetadata = $productMetadata;
        $this->productRepository = $productRepository;
        $this->collectionFactory = $collectionFactory;
        $this->websiteFactory = $websiteFactory;
        $this->collectionTaxFactory = $collectionTaxFactory;
        $this->importCollection = $importCollectionFactory;
        $this->_specialAttributes[] = '_tier_price_value_type';
        $this->_fieldsMap += [
            '_tier_price_website' => 'tier_price_website',
            '_tier_price_customer_group' =>
'tier_price_customer_group',
            '_tier_price_qty' => 'tier_price_qty',
            '_tier_price_price' => 'tier_price_price',
            '_tier_price_value_type' => 'tier_price_value_type',
        ];
        $this->_isRowCategoryMapped = false;
        $this->manager = $moduleManager;
        $this->urlKeyManager = $urlKeyManager;
        $this->productActionFactory = $productActionFactory;
        $this->categoryLinkRepository = $categoryLinkRepository;
        $this->productCollectionFactory = $collectionFactory;
        foreach ($integrations as $integration) {
            if (!($integration instanceof IntegrationInterface)) {
                throw new LocalizedException(__(
                    'Integration class must be instance of
"%interface"',
                    ['interface' => IntegrationInterface::class]
                ));
            }
        }
        $this->integrations = $integrations;
        $this->_fireConstruct();
    }

    /**
     * @return string
     * @throws Exception
     */
    public function _getProductEntityLinkField()
    {
        return $this->getProductEntityLinkField();
    }

    /**
     * @return $this
     * @throws Exception
     */
    public function _fireConstruct()
    {
        $this->urlKeyManager
            ->setEntity($this)
            ->initUrlKeys();
        return $this;
    }

    /**
     * Remove large objects
     */
    public function __destruct()
    {
        unset($this->_optionEntity);
    }

    /**
     * import product data
     *
     * @return bool
     * @throws Exception
     * @throws LocalizedException
     */
    public function importData()
    {
        $this->notValidedSku = [];
        if ($this->_parameters['behavior'] ==
Import::Ivan_ONLY_UPDATE) {
            $this->onlyUpdate = 1;
            $this->_parameters['behavior'] = Import::BEHAVIOR_APPEND;
        } elseif ($this->_parameters['behavior'] ==
Import::Ivan_ONLY_ADD) {
            $this->onlyAdd = true;
            $this->_parameters['behavior'] = Import::BEHAVIOR_APPEND;
        }
        $this->_validatedRows = null;

        if (Import::BEHAVIOR_REPLACE == $this->getBehavior()) {
            $this->_replaceFlag = true;
            $this->replaceProducts();
        } elseif (Import::BEHAVIOR_DELETE == $this->getBehavior()) {
            $this->_deleteProducts();
        } else {
            $this->saveProductsData();
        }
        $this->_eventManager-
>dispatch('catalog_product_import_finish_before', ['adapter' =>
$this]);

        return true;
    }

    /**
```

```
     * Delete products.
     *
     * @return $this
     * @throws Exception
     */
    protected function _deleteProducts()
    {
        $productEntityTable = $this->_resourceFactory->create()->getEntityTable();

        while ($bunch = $this->_dataSourceModel->getNextBunch()) {
            $idsToDelete = [];
            foreach ($bunch as $rowNum => $rowData) {
                $validate = $this->validateRow($rowData, $rowNum);
                $scopeStore = self::SCOPE_STORE == $this->getRowScope($rowData);
                $scopeDefault = self::SCOPE_DEFAULT == $this->getRowScope($rowData);
                if ($validate && ($scopeStore || $scopeDefault)) {
                    $oldSku = $this->getExistingSku($rowData[self::COL_SKU])['entity_id'];
                    if (!empty($oldSku)) {
                        $idsToDelete[] = $oldSku;
                    }
                }
            }
            if ($idsToDelete) {
                $this->countItemsDeleted += count($idsToDelete);
                $this->_processedEntitiesCount += count($idsToDelete);
                $this->transactionManager->start($this->_connection);
                try {
                    $this->objectRelationProcessor->delete(
                        $this->transactionManager,
                        $this->_connection,
                        $productEntityTable,
                        $this->_connection->quoteInto('entity_id IN (?)', $idsToDelete),
                        ['entity_id' => $idsToDelete]
                    );
                    $this->_eventManager->dispatch(
                        'catalog_product_import_bunch_delete_commit_before',
                        [
                            'adapter' => $this,
                            'bunch' => $bunch,
                            'ids_to_delete' => $idsToDelete,
                        ]
                    );
                    $this->transactionManager->commit();
                } catch (Exception $e) {
                    $this->transactionManager->rollBack();
                    throw $e;
                }
                $this->_eventManager->dispatch(
                    'catalog_product_import_bunch_delete_after',
                    ['adapter' => $this, 'bunch' => $bunch]
                );
            }
        }
        return $this;
    }

    /**
     * Replace imported products.
     *
     * @return $this
     * @throws LocalizedException
     * @throws Zend_Validate_Exception
     */
    protected function replaceProducts()
    {
        $this->deleteProductsForReplacement();
        $this->_oldSku = $this->skuProcessor->reloadOldSkus()->getOldSkus();
        $this->_validatedRows = null;
        $this->setParameters(
            array_merge(
                $this->getParameters(),
                ['behavior' => Import::BEHAVIOR_APPEND]
            )
        );
        $this->saveProductsData();

        return $this;
    }

    /**
     * find url_key duplicates
     *
     * @param array $bunchRows
     * @return array $bunchRows
     */
    protected function findUrlKeyDuplicates($bunchRows)
    {
        $urlKeys = [];
        foreach ($bunchRows as $rowNum => $rowData) {
            if (array_key_exists(self::COL_STORE_VIEW_CODE, $rowData)
                && $rowData[self::COL_STORE_VIEW_CODE] === null
            ) {
                $storeCode = 'admin';
            } else {
                $storeCode = $rowData[self::COL_STORE_VIEW_CODE] ?? 'default';
            }
            if (!isset($urlKeys[$storeCode])) {
                $urlKeys[$storeCode] = [];
            }

            if (isset($rowData[self::URL_KEY]) && in_array($rowData[self::URL_KEY], $urlKeys[$storeCode])) {
                unset($bunchRows[$rowNum]);
                $message = 'product with sku: %1 not imported because its url is not unique.';
                $this->addLogWriteln(__($message, $this->getCorrectSkuAsPerLength($rowData)), $this->output, 'info');
            } else {
                $urlKeys[$storeCode][] = $rowData[self::URL_KEY] ?? '';
            }
        }

        return $bunchRows;
    }

    /**
     * Save products data.
     *
     * @return $this
     * @throws LocalizedException
     * @throws Zend_Validate_Exception
     */
    protected function saveProductsData()
    {
        foreach ($this->_productTypeModels as $productTypeModel) {
            if ($productTypeModel instanceof Downloadable) {
                $productTypeModel->clearObject();
            }
        }
        $this->saveProducts();
        foreach ($this->_productTypeModels as $productTypeModel) {
            $productTypeModel->saveData();
        }

        $this->_saveLinks();
        $this->_saveStockItem();

        if ($this->_replaceFlag) {
            $this->getOptionEntity()->clearProductsSkuToId();
        }
        $this->getOptionEntity()->importData();
        $verbosity = false;
        if (!$this->helper->getProcessor()->inConsole) {
            $verbosity = ConsoleOutput::VERBOSITY_VERBOSE;
        }
        if (is_array($this->integrations)) {
            /**
             * @var $moduleKey
             * @var Import\Product\Integration\AbstractIntegration $integration
             */
            foreach ($this->integrations as $moduleKey => $integration) {
                if ($this->manager->isEnabled($moduleKey)) {
                    $integration->setAdapter($this);
                    $integration->setDataSourceModel($this->_dataSourceModel);
                    $integration->importData($verbosity);
                }
            }
        }
        foreach ($this->productLinkData as $idConfigProduct => $typesLink) {
            $this->addProductLinks($idConfigProduct, $typesLink);
        }

        return $this;
    }

    /**
     * Save Stock Item.
     *
     * @return $this
     * @throws LocalizedException
     */
    protected function _saveStockItem()
    {
        /** @var $stockResource Item */
        $stockResource = $this->_stockResItemFac->create();
        $entityTable = $stockResource->getMainTable();
        while ($bunch = $this->_dataSourceModel->getNextBunch()) {
            $stockData = [];
            // Format bunch to stock data rows
            foreach ($bunch as $rowNum => $rowData) {
                if (!$this->isRowAllowedToImport($rowData, $rowNum)) {
                    continue;
                }

                if (isset($this->_parameters['increase_product_stock_by_qty'])
                    && $this->_parameters['increase_product_stock_by_qty'] == 1
                ) {
                    $rowData['qty'] = (int)($rowData['qty'] + $this->getProductStockQty($rowData[self::COL_SKU]));
                }

                $sku = $rowData[self::COL_SKU];
```

```
                $row = [];
                if ($this->skuProcessor->getNewSku($sku) !== null) {
                    $row = $this->formatStockDataForRow($rowData);
                }

                if (!isset($stockData[$sku])) {
                    $stockData[$sku] = $row;
                }
            }

            // Insert rows
            if (!empty($stockData)) {
                if ($this->stockItemImporter instanceof
StockItemImporterInterface
                    && version_compare($this->productMetadata-
>getVersion(), '2.3.0', '>=')
                ) {
                    try {
                        $this->stockItemImporter->import($stockData);
                    } catch (Exception $exception) {
                        $this->addLogWriteln($exception-
>getMessage(), $this->getOutput(), 'info');
                        $this->getLogger()->debug($exception);
                    }
                } else {
                    $this->_connection-
>insertOnDuplicate($entityTable, array_values($stockData));
                }
            }
        }
        return $this;
    }

    /**
     * Format row data to DB compatible values
     *
     * @param array $rowData
     * @return array
     * @throws Exception
     */
    private function formatStockDataForRow(array $rowData)
    {
        $sku = $rowData[self::COL_SKU];
        $row['product_id'] = $this->skuProcessor-
>getNewSku($sku)['entity_id'];
        $row['website_id'] = $this->stockConfiguration-
>getDefaultScopeId();
        $row['stock_id'] = $this->stockRegistry-
>getStock($row['website_id'])->getStockId();

        /** @var \Magento\CatalogInventory\Model\Stock\Item
$stockItemDo */
        $stockItemDo = $this->stockRegistry-
>getStockItem($row['product_id'], $row['website_id']);
        $existStockData = $stockItemDo->getData();

        $row = array_merge(
            $this->defaultStockData,
            array_intersect_key($existStockData, $this-
>defaultStockData),
            array_intersect_key($rowData, $this->defaultStockData),
            $row
        );

        $typeId = $this->skuProcessor->getNewSku($sku)['type_id'];
        if ($this->stockConfiguration->isQty($typeId)) {
            if ($this->manager-
>isEnabled('Magento_CatalogMessageBus')) {
                if (isset($existStockData['qty'])) {
                    $row['qty'] = $existStockData['qty'];
                }
            }
            $stockItemDo->setData($row);
            $row['is_in_stock'] = $stockItemDo->getBackorders() &&
isset($row['is_in_stock'])
                ? $row['is_in_stock']
                : $this->stockStateProvider-
>verifyStock($stockItemDo);
            if ($this->stockStateProvider-
>verifyNotification($stockItemDo)) {
                $row['low_stock_date'] = $this->dateTime->gmDate(
                    'Y-m-d H:i:s',
                    (new \DateTime())->getTimestamp()
                );
            }
            $row['stock_status_changed_auto'] =
                (int)!$this->stockStateProvider-
>verifyStock($stockItemDo);
        } else {
            $row['qty'] = 0;
        }

        return $row;
    }

    /**
     * get product qty
     *
     * @param string $sku
     *
     * @return float
     */
    protected function getProductStockQty($sku)
    {
        try {
            $stockItem = $this->stockRegistry-
>getStockItemBySku($sku);
            $qty = $stockItem->getData('qty');
        } catch (NoSuchEntityException $e) {
            $qty = 0;
        }

        return $qty;
    }

    /**
     * Set valid attribute set and product type to rows with all
scopes
     * to ensure that existing products doesn't changed.
     *
     * @param array $rowData
     *
     * @return array
     */
    protected function _prepareRowForDb(array $rowData)
    {
        $productType = isset($rowData[self::COL_TYPE]) ?
$rowData[self::COL_TYPE] : '';
        $rowData = parent::_prepareRowForDb($rowData);
        if ($productType) {
            $rowData[self::COL_TYPE] = $productType;
        }
        if (!$this->onlyUpdate) {
            foreach ($this->defaultStockData as $key => $value) {
                if (isset($rowData[$key])) {
                    if ($rowData[$key] === true) {
                        $rowData[$key] = 1;
                    } elseif ($rowData[$key] === false) {
                        $rowData[$key] = 0;
                    } elseif ($rowData[$key] === '') {
                        $rowData[$key] = 0;
                    } elseif ($rowData[$key] === null && $key !=
'low_stock_date') {
                        $rowData[$key] = 0;
                    }
                } else {
                    if ($key == 'qty') {
                        $rowData[$key] = $this-
>getProductStockQty($rowData[self::COL_SKU]);
                    } else {
                        $rowData[$key] = $value;
                    }
                }
            }
        }
        $rowData = $this->adjustBundleTypeAttributes($rowData);
        return $rowData;
    }

    /**
     * @param array $rowData
     * @param array $configurableData
     */
    protected function prepareConfigurableVariation(array $rowData,
array &$configurableData = [])
    {
        $confSwitch = $this->_parameters['configurable_switch'];

        if ($confSwitch && isset($rowData['product_type'])
            && ($rowData['product_type'] == 'simple' ||
$rowData['product_type'] == 'virtual')) {
            $field = $this->_parameters['configurable_field'];
            $skuConf = null;
            if (isset($rowData[$field])) {
                switch ($this->_parameters['configurable_type']) {
                    case TypeOptions::FIELD:
                        if ($rowData[$field] && $this-
>getCorrectSkuAsPerLength($rowData) != $rowData[$field]) {
                            $skuConf = $rowData[$field];
                        }
                        break;
                    case TypeOptions::PART_UP:
                        $array = explode($this-
>_parameters['configurable_part'], $rowData[$field]);
                        if (count($array) > 1) {
                            $skuConf = $array[0];
                        }
                        break;
                    case TypeOptions::PART_DOWN:
                        $array = explode($this-
>_parameters['configurable_part'], $rowData[$field]);
                        if (count($array) > 1) {
                            $skuConf = $array[count($array) - 1];
                        }
                        break;
                    case TypeOptions::SUB_UP:
                        $skuConf = substr($rowData[$field], 0, $this-
>_parameters['configurable_symbols']);
                        break;
                    case TypeOptions::SUB_DOWN:
                        $skuConf = substr($rowData[$field], -$this-
>_parameters['configurable_symbols']);
                        break;
                }
            }
            if ($this->_replaceFlag && !isset($this-
>_oldSku[mb_strtolower($skuConf)])) {
                $skuConf = null;
            }
            if ($skuConf) {
```

```
                $newData = $rowData;
                $arrayConf = [];
                if (!empty($this->_parameters['configurable_variations'])) {
                    foreach ($this->_parameters['configurable_variations'] as $attrField) {
                        if (isset($newData[$attrField]) && trim($newData[$attrField]) !== '') {
                            $arrayConf[$attrField] = $newData[$attrField];
                        }
                    }
                }
                if (!empty($arrayConf)) {
                    $arrayConf['sku'] = (string) $newData['sku'];
                    $configurableData[(string) $skuConf][] = $arrayConf;
                }
            }
        }
    }

    /**
     * Gather and save information about product entities.
     *
     * @return $this
     * @throws LocalizedException
     * @throws Zend_Validate_Exception
     * @throws Exception
     * @SuppressWarnings(PHPMD.CyclomaticComplexity)
     * @SuppressWarnings(PHPMD.NPathComplexity)
     * @SuppressWarnings(PHPMD.ExcessiveMethodLength)
     */
    protected function saveProducts()
    {
        $this->translator = $this->translator->init($this->_parameters);
        $existingImages = [];
        $existingUpload = [];
        $entityLinkField = $this->getProductEntityLinkField();
        if (!empty($this->_parameters['import_source']) && $this->_parameters['import_source'] != 'file') {
            $this->_initSourceType($this->_parameters['import_source']);
        }
        $configurableData = [];

        $isPriceGlobal = $this->_catalogData->isPriceGlobal();
        $productLimit = null;
        $productsQty = null;
        $this->importImage->setConfig($this->_parameters);

        while ($nextBunch = $this->_dataSourceModel->getNextBunch()) {
            $entityRowsIn = $entityRowsUp = [];
            $attributes = [];
            $this->websitesCache = $this->categoriesCache = $this->categoryProductPosition = [];
            $this->categoryProcessor->setRowCategoryPosition($this->categoryProductPosition);
            $mediaGallery = $uploadedImages = [];
            $tierPrices = [];
            $previousType = $prevAttributeSet = null;
            $existingImages = $this->getExistingImages($nextBunch);
            if ($this->sourceType && $this->_parameters['image_import_source']) {
                $nextBunch = $this->prepareImagesFromSource($nextBunch);
            }

            $prevData = [];
            $createValuesAllowed = (bool)$this->scopeConfig->getValue(
                Import::CREATE_ATTRIBUTES_CONF_PATH,
                ScopeInterface::SCOPE_STORE
            );
            $storeIds = $this->getStoreIds();
            foreach ($nextBunch as $rowNum => $rowData) {
                $time = explode(" ", microtime());
                $startTime = $time[0] + $time[1];
                if (isset($rowData[self::COL_CATEGORY])) {
                    $categoriesMapping = $this->categoriesMapping($rowData);
                    $rowData[self::COL_CATEGORY] = $categoriesMapping[self::COL_CATEGORY];

                    if (!empty($categoriesMapping[self::COL_CATEGORY . '_position'])) {
                        $rowData[self::COL_CATEGORY . '_position'] =
                            $categoriesMapping[self::COL_CATEGORY . '_position'];
                    }
                }

                $rowData = $this->joinIdenticalyData($rowData);
                if (isset($rowData['_attribute_set']) && isset($rowData['attribute_set_code'])) {
                    if (isset($rowData['update_attribute_set']) && ((int)($rowData['update_attribute_set']) > 0)) {
                        $rowData['_attribute_set'] = $rowData['attribute_set_code'];
                    } else {
                        unset($rowData['attribute_set_code']);
                    }
                }

                $oldSkus = $this->skuProcessor->getOldSkus();
                $sku = strtolower($this->getCorrectSkuAsPerLength($rowData));

                if (!isset($oldSkus[$sku])) {
                    if (!isset($rowData['_attribute_set'])
                        || (isset($rowData['_attribute_set']) && empty($rowData['_attribute_set']))
                    ) {
                        $collectSets = $this->_attrSetIdToName;
                        reset($collectSets);
                        $rowData['_attribute_set'] = current($collectSets);
                    }

                    if (isset($this->_parameters['remove_related_product'])
                        && $this->_parameters['remove_related_product'] == 1
                    ) {
                        $this->removeRelatedProducts($this->getCorrectSkuAsPerLength($rowData));
                    }

                    if (isset($this->_parameters['remove_crosssell_product'])
                        && $this->_parameters['remove_crosssell_product'] == 1
                    ) {
                        $this->removeCrosssellProducts($this->getCorrectSkuAsPerLength($rowData));
                    }

                    if (isset($this->_parameters['remove_upsell_product'])
                        && $this->_parameters['remove_upsell_product'] == 1
                    ) {
                        $this->removeUpsellProducts($this->getCorrectSkuAsPerLength($rowData));
                    }

                    $rowData = $this->checkAdditionalImages($rowData);
                    $rowData = $this->customChangeData($rowData);

                    if (!$this->validateRow($rowData, $rowNum) || !$this->validateRowByProductType($rowData, $rowNum)) {
                        $this->addLogWriteln(
                            __('product with sku: %1 is not valided', $this->getCorrectSkuAsPerLength($rowData)),
                            $this->output,
                            'info'
                        );
                        $this->notValidedSku[] = strtolower($this->getCorrectSkuAsPerLength($rowData));
                        unset($nextBunch[$rowNum]);
                        continue;
                    } else {
                        $rowData = $this->stripSlashes($rowData);
                    }

                    $productType = isset($rowData[self::COL_TYPE]) ?
                        strtolower($rowData[self::COL_TYPE]) :
                        $this->skuProcessor->getNewSku($this->getCorrectSkuAsPerLength($rowData))['type_id'];
                    // custom
                    if ($productType) {
                        $productTypeModel = $this->_productTypeModels[$productType];
                        if ($createValuesAllowed) {
                            $rowData = $this->createAttributeValues(
                                $productTypeModel,
                                $rowData
                            );
                        }
                    }

                    if (!isset($rowData[self::COL_ATTR_SET]) ||
                        !isset($this->_attrSetNameToId[$rowData[self::COL_ATTR_SET]])) {
                        $this->addRowError(ValidatorInterface::ERROR_INVALID_ATTR_SET, $rowNum);
                        $this->addLogWriteln(
                            __(
                                'product with sku: %1 is not valided. ' .
                                'Invalid value for Attribute Set column (set doesn\'t exist?)',
                                $this->getCorrectSkuAsPerLength($rowData)
                            ),
                            $this->output,
                            'info'
                        );
                        $this->notValidedSku[] = strtolower($this->getCorrectSkuAsPerLength($rowData));
                        unset($nextBunch[$rowNum]);
                        continue;
                    }

                    $urlKey = null;
                    if (!($this->onlyUpdate && empty($rowData[self::URL_KEY]))) {
                        $urlKey = $this->getUrlKey($rowData);
                    }
```

```
                if ($urlKey) {
                    if (!empty($rowData[self::URL_KEY])) {
                        // If url_key column and its value were in
the CSV file
                        $rowData[self::URL_KEY] = $urlKey;
                    } elseif ($this->isNeedToChangeUrlKey($rowData))
{
                        // If url_key column was empty or even not
declared in the CSV file but by the rules
                        // it is need to be setteed. In case when
url_key is generating from name column we
                        // have to ensure that the bunch of products
will pass for the event with url_key column.
                        $nextBunch[$rowNum][self::URL_KEY] =
$rowData[self::URL_KEY] = $urlKey;
                    } elseif (isset($rowData[self::URL_KEY]) ||
isset($rowData[self::COL_NAME])) {
                        $rowData[self::URL_KEY] = $urlKey;
                    }
                }

                $this->urlKeys = [];
                $rowData = $this-
>applyCategoryLevelSeparator($rowData);
                $rowData = $this-
>adjustBundleTypeAttributes($rowData);

                if (empty($this->getCorrectSkuAsPerLength($rowData)))
{
                    $rowData = array_merge($prevData, $this-
>deleteEmpty($rowData));
                } else {
                    $prevData = $rowData;
                }
                $sku = $this->getCorrectSkuAsPerLength($rowData);
                if ($this->onlyUpdate) {
                    $collectionUpdate = $this->collectionFactory-
>create()->addFieldToFilter(
                        self::COL_SKU,
                        $this->getCorrectSkuAsPerLength($rowData)
                    );
                    if (!$collectionUpdate->getSize()) {
                        $this->addLogWriteln(__('product with sku: %1
does not exist', $sku), $this->output, 'info');
                        unset($nextBunch[$rowNum]);
                        continue;
                    }
                }
                if ($this->getErrorAggregator()->hasToBeTerminated())
{
                    $this->getErrorAggregator()-
>addRowToSkip($rowNum);
                    unset($nextBunch[$rowNum]);
                    $this->notValidedSku[] = strtolower($this-
>getCorrectSkuAsPerLength($rowData));

                    continue;
                }

                if (isset($rowData['_attribute_set']) && isset($this-
>_attrSetNameToId[$rowData['_attribute_set']])) {
                    $this->skuProcessor->setNewSkuData(
                        $this->getCorrectSkuAsPerLength($rowData),
                        'attr_set_id',
                        $this-
>_attrSetNameToId[$rowData['_attribute_set']]
                    );
                }

                $this->prepareConfigurableVariation($rowData,
$configurableData);
                $rowScope = $this->getRowScope($rowData);
                $rowSku = $this->getCorrectSkuAsPerLength($rowData);
                $checkSku = $rowSku;

                if (version_compare($this->productMetadata-
>getVersion(), '2.2.0', '>=')) {
                    $checkSku = strtolower($rowSku);
                }
                if (!$rowSku) {
                    $this->getErrorAggregator()-
>addRowToSkip($rowNum);
                    continue;
                } elseif (self::SCOPE_STORE == $rowScope) {
                    // set necessary data from SCOPE_DEFAULT row
                    $rowData[self::COL_TYPE] = $this->skuProcessor-
>getNewSku($checkSku)['type_id'];
                    $rowData['attribute_set_id'] = $this-
>skuProcessor->getNewSku($checkSku)['attr_set_id'];
                    $rowData[self::COL_ATTR_SET] = $this-
>skuProcessor->getNewSku($checkSku)['attr_set_code'];
                }

                // Entity phase
                if (!isset($this->_oldSku[$checkSku])) {
                    // new row
                    if (!$productLimit || $productsQty <
$productLimit) {
                        if (isset($rowData['has_options'])) {
                            $hasOptions = $rowData['has_options'];
                        } else {
                            $hasOptions = 0;
                        }
                        $entityRowsIn[$rowSku] = [
                            'attribute_set_id' => $this-
>skuProcessor->getNewSku($checkSku)['attr_set_id'],
                            'type_id' => $this->skuProcessor-
>getNewSku($checkSku)['type_id'],
                            'sku' => $rowSku,
                            'has_options' => $hasOptions,
                            'created_at' => $this->_localeDate-
>date()->format(DateTime::DATETIME_PHP_FORMAT),
                            'updated_at' => $this->_localeDate-
>date()->format(DateTime::DATETIME_PHP_FORMAT),
                        ];
                        $productsQty++;
                    } else {
                        $rowSku = null;
                        // sign for child rows to be skipped
                        $this->getErrorAggregator()-
>addRowToSkip($rowNum);
                        continue;
                    }
                } else {
                    $array = [
                        'updated_at' => $this->_localeDate->date()-
>format(DateTime::DATETIME_PHP_FORMAT),
                        $entityLinkField => $this-
>_oldSku[$checkSku][$entityLinkField],
                    ];
                    $array['attribute_set_id'] = $this->skuProcessor-
>getNewSku($checkSku)['attr_set_id'];
                    $array['type_id'] = $productType;
                    // existing row
                    $entityRowsUp[] = $array;
                }

                // Categories phase
                if (!array_key_exists($rowSku, $this-
>categoriesCache)) {
                    $this->categoriesCache[$rowSku] = [];
                }

                $rowData['rowNum'] = $rowNum;
                $categoryIds = $this->getCategories($rowData);
                if (isset($rowData['category_ids'])) {
                    $catIds = explode($this-
>getMultipleValueSeparator(), $rowData['category_ids']);
                    $finalCatId = [];
                    foreach ($catIds as $catId) {
                        $catId = (int)$catId;
                        $existingCat = $this->categoryProcessor-
>getCategoryById($catId);
                        if (is_int($catId) && $catId > 0 &&
$existingCat && $existingCat->getId()) {
                            $finalCatId[] = $catId;
                        }
                    }
                    $categoryIds = array_merge($categoryIds,
$finalCatId);
                }
                if (!empty($this->getCategoryProcessor()-
>getFailedCategories())) {
                    foreach ($this->getCategoryProcessor()-
>getFailedCategories() as $field) {
                        $this->addRowError('Category: ' .
__($field['category']->getName() .
                            ' Url: ' . $field['category']-
>getUrlKey() .
                            ' ' . $field['exception']-
>getMessage()), $rowNum);
                    }
                    $this->addLogWriteln(
                        __('product with sku: %1 is not valided',
$this->getCorrectSkuAsPerLength($rowData)),
                        $this->output,
                        'info'
                    );
                    $this->notValidedSku[] = strtolower($this-
>getCorrectSkuAsPerLength($rowData));
                    unset($nextBunch[$rowNum]);
                    continue;
                }
                foreach ($categoryIds as $id) {
                    $this->categoriesCache[$rowSku][$id] = true;
                }

                $catIds = [];
                if ($this->isSkuExist($rowSku)) {
                    if (!isset($this-
>_oldSku[strtolower($rowData[self::COL_SKU])]['entity_id'])) {
                        $entityId = $this-
>_oldSku[strtolower($rowData[self::COL_SKU])]['row_id'];
                        $this->skuProcessor-
>setNewSkuData($rowData[self::COL_SKU], 'entity_id', $entityId);
                    } else {
                        $entityId = $this-
>_oldSku[strtolower($rowData[self::COL_SKU])]['entity_id'];
                    }
                    $oldCategoryIds = $this-
>getCategoryLinks($entityId);

                    if (!empty($categoryIds)
                        && isset(
                            $this-
>_parameters['remove_product_categories'],
                            $this-
>_oldSku[strtolower($rowData[self::COL_SKU])]
                        )
```

```
                        && $this-
>_parameters['remove_product_categories'] > 0
                    ) {
                        foreach ($oldCategoryIds as $oldCategoryId) {
                            if
(!in_array($oldCategoryId['category_id'], $categoryIds, false)) {
                                $this-
>categoriesCache[$rowSku][$oldCategoryId['category_id']] = false;
                            }
                        }
                    }
                }

                if (!isset($this->categoryProductPosition[$rowSku]))
{
                    $this->categoryProductPosition[$rowSku] = [];
                }
                $this->categoryProductPosition[$rowSku] += $this-
>categoryProcessor->getRowCategoryPosition();

                if (isset($rowData[self::COL_CATEGORY]) &&
empty($rowData[self::COL_CATEGORY])) {
                    foreach ($catIds as $categoryId) {
                        $this->categoryLinkRepository-
>deleteByIds($categoryId, $rowData[self::COL_SKU]);
                        $this->categoriesCache[$rowSku] = [];
                        $this->categoryProductPosition[$rowSku] = [];
                    }
                }

                unset($rowData['rowNum']);
                if (!array_key_exists($rowSku, $this->websitesCache))
{
                    $this->websitesCache[$rowSku] = [];
                }
                // Product-to-Website phase
                if (!empty($rowData[self::COL_PRODUCT_WEBSITES])) {
                    $websiteCodes = explode($this-
>getMultipleValueSeparator(), $rowData[self::COL_PRODUCT_WEBSITES]);
                    foreach ($websiteCodes as $websiteCode) {
                        $websiteId = $this->storeResolver-
>getWebsiteCodeToId($websiteCode);
                        $this->websitesCache[$rowSku][$websiteId] =
true;
                    }
                }
                // Price rules
                $rowData = $this->applyPriceRules($rowData);
                $fixedName = __("Fixed");
                $fixed = $fixedName;
                if (isset($rowData['_tier_price_value_type'])) {
                    $fixed = $rowData['_tier_price_value_type'] ==
$fixedName;
                }
                // Tier prices phase
                if (!empty($rowData['_tier_price_website'])) {
                    $tierPrices[$rowSku][] = [
                        'all_groups' =>
$rowData['_tier_price_customer_group'] == self::VALUE_ALL,
                        'customer_group_id' =>
$rowData['_tier_price_customer_group'] ==
                        self::VALUE_ALL ? 0 :
$rowData['_tier_price_customer_group'],
                        'qty' => $rowData['_tier_price_qty'],
                        'value' => ($fixed) ?
$rowData['_tier_price_price'] : 0,
                        'website_id' => self::VALUE_ALL ==
$rowData['_tier_price_website'] || $isPriceGlobal
                            ? 0
                            : $this->storeResolver-
>getWebsiteCodeToId(
                                $rowData['_tier_price_website']
                            ),
                        'percentage_value' => (!$fixed) ?
$rowData['_tier_price_price'] : 0,
                    ];
                    $tierPrices = array_merge($tierPrices, $this-
>getTierPrices($rowData, $rowSku));
                } else {
                    $tierPrices += $this->getTierPrices($rowData,
$rowSku);
                }
                if (!$this->validateRow($rowData, $rowNum)) {
                    $this->addLogWriteln(__('product with sku: %1 is
not valided', $sku), $this->output, 'info');
                    unset($nextBunch[$rowNum]);
                    continue;
                }
                // Media gallery phase
                if ($this->publisher && isset($this-
>_parameters['deferred_images']) &&
                    $this->_parameters['deferred_images']) {
                    $this->importImage->addMediaGalleryRows(
                        $rowData,
                        $mediaGallery,
                        $existingImages,
                        $uploadedImages,
                        $rowNum
                    );
                } else {
                    $this->importImageProcessor->setConfig($this-
>_parameters);
                    $this->processMediaGalleryRows(
                        $rowData,
                        $mediaGallery,
                        $existingImages,
                        $uploadedImages,
                        $rowNum
                    );
                }

                if (!$productType === null) {
                    $previousType = $productType;
                }
                $prevAttributeSet = null;
                if (isset($rowData[self::COL_ATTR_SET])) {
                    $prevAttributeSet = $rowData[self::COL_ATTR_SET];
                }
                if (self::SCOPE_NULL == $rowScope) {
                    // for multiselect attributes only
                    if (!$prevAttributeSet === null) {
                        $rowData[self::COL_ATTR_SET] =
$prevAttributeSet;
                    }
                    if ($productType === null && !$previousType ===
null) {
                        $productType = $previousType;
                    }
                    if ($productType === null) {
                        continue;
                    }
                }
                if (!$productType) {
                    $tempProduct = $this->skuProcessor-
>getNewSku($checkSku);
                    if (isset($tempProduct['type_id'])) {
                        $productType = $tempProduct['type_id'];
                    }
                }
                if ($productType) {
                    $rowScope = empty($rowData[self::COL_STORE]) ?
self::SCOPE_DEFAULT : self::SCOPE_STORE;
                    $rowStore = (self::SCOPE_STORE == $rowScope)
                        ? $this->storeResolver-
>getStoreCodeToId($rowData[self::COL_STORE])
                        : 0;
                    $productTypeModel = $this-
>_productTypeModels[$productType];

                    if (!empty($rowData['tax_class_name'])) {
                        $rowData['tax_class_name'] = $this-
>getCurrentTaxClass($rowData['tax_class_name']);
                        $rowData['tax_class_id'] =
                            $this->taxClassProcessor-
>upsertTaxClass($rowData['tax_class_name'], $productTypeModel);
                    }

                    if ($this->getBehavior() ==
Import::BEHAVIOR_APPEND ||
                        empty($this-
>getCorrectSkuAsPerLength($rowData))) {
                        if (isset($this-
>_parameters['clear_attribute_value'])
                            && $this-
>_parameters['clear_attribute_value'] == 0) {
                            $rowData = $productTypeModel-
>clearEmptyData($rowData);
                        }
                    }

                    if (isset($this-
>_parameters['clear_attribute_value'])
                        && $this-
>_parameters['clear_attribute_value'] == 1
                    ) {
                        $rowData[self::COL_STORE] = null;
                    }
                    $rowData = $productTypeModel-
>prepareAttributesWithDefaultValueForSave(
                        $rowData,
                        !isset($this->_oldSku[$checkSku])
                    );
                    //google translation data
                    if ($this->translator->isTranslatorSet()) {
                        $translateAttributes = $this-
>_parameters['translate_attributes'] ?? [];
                        $translateStore = (int)($this-
>_parameters['translate_store_ids'] ?? 0);
                    }

                    foreach ($rowData as $attrCode => $attrValue) {
                        $attribute = $this-
>retrieveAttributeByCode($attrCode);
                        if ('multiselect' != $attribute-
>getFrontendInput() && self::SCOPE_NULL == $rowScope) {
                            // skip attribute processing for
SCOPE_NULL rows
                            continue;
                        }
                        $attrId = $attribute->getId();
                        $backModel = $attribute->getBackendModel();
                        $attrTable = $attribute->getBackend()-
>getTable();
                        $storeIds = [0];

                        if ('datetime' == $attribute-
>getBackendType()
                            && (
                                in_array($attribute-
>getAttributeCode(), $this->dateAttrCodes)
```

```
                                || $attribute->getIsUserDefined()
                            )
                        ) {
                            $attrValue = $this->dateTime-
>formatDate($attrValue, false);
                        } elseif ('datetime' == $attribute-
>getBackendType() && strtotime($attrValue)) {
                            $attrValue = gmdate(
                                'Y-m-d H:i:s',
                                $this->_localeDate->date($attrValue)-
>getTimestamp()
                            );
                        }

                        $defaultValue = $this-
>getDefaultAttrValue($attribute, $rowSku);
                        $storeValue = $this-
>getDefaultAttrValue($attribute, $rowSku, $rowStore);

                        if (!isset($this-
>adminAttributeValue[$rowSku])) {
                            $this->adminAttributeValue = [$rowSku =>
[]];
                        }

                        if (false === $defaultValue && $rowStore ==
0) {
                            $this-
>adminAttributeValue[$rowSku][$attrCode] = $attrValue;
                        }

                        /*
                         * If storeValue exists and the default value
is same as new value then remove it
                         */
                        if ($storeValue && $defaultValue ===
(string)$attrValue && $rowStore > 0) {
                            $this-
>_deleteStoreAttributeValue($attribute, $rowSku, $rowStore);
                        }
                        if ($this->translator->isTranslatorSet()) {
                            if (!empty($translateAttributes)
                                && !empty($translateStore)
                                &&
!isset($attributes[$attrTable][$rowSku][$attrId][$translateStore])
                                && in_array($attrCode,
$translateAttributes, true)
                            ) {
                                $storeValue = $this->translator
                                    -
>translateAttributeValue($attrValue, $attrCode, $translateStore);

$attributes[$attrTable][$rowSku][$attrId][$translateStore] =
$storeValue;
                            } elseif (isset($translateStore) &&
$translateStore > 0) {
                                $this-
>_deleteStoreAttributeValue($attribute, $rowSku, $translateStore);
                            }
                        }
                        if ($defaultValue && ($defaultValue ===
(string)$attrValue)) {
                            continue;
                        }

                        $adminValue = $this-
>adminAttributeValue[$rowSku][$attrCode] ?? false;
                        if (false !== $adminValue && $adminValue ===
$attrValue && $rowStore > 0) {
                            continue;
                        }

                        if (self::SCOPE_STORE == $rowScope) {
                            if (self::SCOPE_WEBSITE == $attribute-
>getIsGlobal()) {
                                // check website defaults already set
                                if
(!isset($attributes[$attrTable][$rowSku][$attrId][$rowStore])) {
                                    $storeIds = $this->storeResolver-
>getStoreIdToWebsiteStoreIds($rowStore);
                                }
                            } elseif (self::SCOPE_STORE ==
$attribute->getIsGlobal() || $attrCode == 'price') {
                                $storeIds = [$rowStore];
                            }

                            if (!isset($this->_oldSku[$checkSku])) {
                                $storeIds[] = 0;
                            }
                        }
                        $storeIds = array_unique($storeIds);
                        sort($storeIds);

                        foreach ($storeIds as $storeId) {
                            if
(!isset($attributes[$attrTable][$rowSku][$attrId][$storeId])) {

$attributes[$attrTable][$rowSku][$attrId][$storeId] = $attrValue;
                            }
                        }
                        // restore 'backend_model' to avoid 'default'
setting
                        $attribute->setBackendModel($backModel);
                    }
```

```
                    $time = explode(" ", microtime());
                    $endTime = $time[0] + $time[1];
                    $totalTime = $endTime - $startTime;
                    $totalTime = round($totalTime, 5);
                    $this->addLogWriteln(__('product with sku: %1
.... %2s', $sku, $totalTime), $this->output, 'info');
                }
            }
            if (method_exists($this, '_saveProductEntity')) {
                $this->_saveProductEntity(
                    $entityRowsIn,
                    $entityRowsUp
                );
            } else {
                $this->saveProductEntity(
                    $entityRowsIn,
                    $entityRowsUp
                );
            }

            $isCached = $this->_parameters['cache_products'] ??
false;
            if ($isCached) {
                $this->saveProductsCache($entityRowsIn,
$entityRowsUp);
            }
            $this->afterSaveNewEntities($entityRowsIn);
            $this->addLogWriteln(__('Imported: %1 rows',
count($entityRowsIn)), $this->output, 'info');
            $this->addLogWriteln(__('Updated: %1 rows',
count($entityRowsUp)), $this->output, 'info');
            $this->_saveProductWebsites(
                $this->websitesCache
            )->_saveProductCategories(
                $this->categoriesCache
            )->_saveProductTierPrices(
                $tierPrices
            );
            if ($this->publisher && isset($this-
>_parameters['deferred_images']) &&
                $this->_parameters['deferred_images']) {
                $this->importImage->publishBranch();
            } else {
                $this->_saveMediaGallery($mediaGallery);
            }
            $this->_saveProductAttributes($attributes);
            $this->_saveProductCategoriesPosition($this-
>categoryProductPosition);

            $this->_eventManager->dispatch(
                'catalog_product_import_bunch_save_after',
                ['adapter' => $this, 'bunch' => $nextBunch]
            );
        }
        if (!empty($configurableData)) {
            $this->saveConfigurationVariations($configurableData,
$existingImages);
        }

        $this->cache->clean([ImportProductCache::BUFF_CACHE]);
        return $this;
    }

    /**
     * @param $inRows
     * @param $upRows
     * @return $this
     */
    protected function saveProductsCache($inRows, $upRows)
    {
        $entityLinkField = $this->getProductEntityLinkField();
        if (!$this->originalImportRows) {
            $this->originalImportRows = $this->cache-
>load(sha1(ImportProductCache::BUFF_CACHE));
            $this->originalImportRows = $this->getSerializer()-
>unserialize($this->originalImportRows) ?? [];
        }

        foreach ($inRows as $row) {
            $id = $this-
>_oldSku[strtolower($row['sku'])]['entity_id'];
            foreach ($this-
>originalImportRows[strtolower($row['sku'])] as $item) {
                $this->cache->save(
                    1,
                    $item,
                    [ImportProductCache::CACHE_TAG . '_' . $id,
ImportProductCache::CACHE_TAG]
                );
            }
        }

        $skuById = array_combine(array_column($this->_oldSku,
'entity_id'), array_keys($this->_oldSku));
        $productIdByRowId = [];
        if ($upRows) {
            $productIdByRowId = isset(current($upRows)['entity_id'])
? [] : $this->getProductIdByRowId($upRows);
        }
        foreach ($upRows as $row) {
            $id = $productIdByRowId ?
$productIdByRowId[$row[$entityLinkField]] : $row['entity_id'];
            $sku = $skuById[$id];
            foreach ($this->originalImportRows[$sku] as $item) {
                $this->cache->save(
```

```
                    1,
                    $item,
                    [ImportProductCache::CACHE_TAG . '_' . $id,
ImportProductCache::CACHE_TAG]
                );
            }
        }

        return $this;
    }

    /**
     * @param $rows
     * @return array
     */
    protected function getProductIdByRowId($rows)
    {
        $select = $this->_connection->select()->from(
            $this->getResource()->getTable('catalog_product_entity'),
            ['row_id', 'entity_id']
        )->where(
            'row_id IN(?)',
            array_column($rows, 'row_id')
        );

        return $this->getProductIdByRowId = $this->_connection-
>fetchPairs($select);
    }

    /**
     * @param $className
     * @return string
     */
    protected function getCurrentTaxClass($className)
    {
        if (!$this->classTaxNames) {
            $select = $this->_connection->select()->from(
                $this->getResource()->getTable('tax_class'),
                'class_name'
            )->where(
                'class_type = ?',
                ClassModel::TAX_CLASS_TYPE_PRODUCT
            );

            $result = [];
            foreach ($this->_connection->fetchCol($select) as $item)
{
                $result [strtolower($item)] = $item;
            }
            $this->classTaxNames = $result;
        }

        $key = strtolower($className);
        return key_exists($key, $this->classTaxNames) ? $this-
>classTaxNames[$key] : $className;
    }

    /**
     * @param string $fileName
     * @param bool $renameFileOff
     * @param array $existingUpload
     *
     * @return string
     */
    protected function uploadMediaFiles($fileName, $renameFileOff =
false, $existingUpload = [])
    {
        return $this->importImageProcessor-
>uploadMediaFiles($fileName, $renameFileOff, $existingUpload);
    }

    /**
     * @param $entityRowsIn
     * @throws Exception
     */
    protected function afterSaveNewEntities($entityRowsIn)
    {
        if ($entityRowsIn) {
            $entityTable = $this->_resourceFactory->create()-
>getEntityTable();

            $select = $this->_connection->select()->from(
                $entityTable,
                array_merge($this->getNewSkuFieldsForSelect(), $this-
>getOldSkuFieldsForSelect())
            )->where(
                $this->_connection->quoteInto('sku IN (?)',
array_keys($entityRowsIn))
            );
            $newProducts = $this->_connection->fetchAll($select);
            foreach ($newProducts as $data) {
                $this->_optionEntity-
>addNewSkuToId($data['entity_id'], $data['sku']);
            }
        }
    }

    /**
     * Return additional data, needed to select.
     * @return array
     */
    private function getOldSkuFieldsForSelect()
    {
        return ['type_id', 'attribute_set_id'];
    }

    /**
     * Get product entity identifier field
     *
     * @return string
     * @throws Exception
     */
    private function getProductIdentifierField()
    {
        if (!$this->productEntityIdentifierField) {
            $this->productEntityIdentifierField = $this-
>getMetadataPool()
                ->getMetadata(ProductInterface::class)
                ->getIdentifierField();
        }
        return $this->productEntityIdentifierField;
    }

    /**
     * Get new SKU fields for select
     *
     * @return array
     * @throws Exception
     */
    private function getNewSkuFieldsForSelect()
    {
        $fields = ['sku', $this->getProductEntityLinkField()];
        if ($this->getProductEntityLinkField() != $this-
>getProductIdentifierField()) {
            $fields[] = $this->getProductIdentifierField();
        }
        return $fields;
    }

    /**
     * @param array $categoriesData
     * @param null $productId
     * @param bool $config
     *
     * @return $this|MagentoProduct
     */
    protected function _saveProductCategories(array $categoriesData,
$productId = null, $config = false)
    {
        static $tableName = null;
        $removeCategories = $this-
>_parameters['remove_product_categories'] ?? 0;
        if ($removeCategories || $productId) {
            if (!$tableName) {
                $tableName = $this->_resourceFactory->create()-
>getProductCategoryTable();
            }
            if ($categoriesData) {
                $categoriesIn = [];
                $delProductId = [];

                foreach ($categoriesData as $productSku =>
$categories) {
                    if (!$config) {
                        $productId = $this->skuProcessor-
>getNewSku($productSku)['entity_id'];
                    }
                    $delProductId[] = $productId;
                    if ($this->onlyUpdate
                        && (int)$removeCategories > 0
                        && empty($categories)
                    ) {
                        $this->addLogWriteln(
                            __('Product %1 Categories Cannot be
Cleared', $productSku),
                            $this->output,
                            'info'
                        );
                        continue;
                    }
                    $cat = [];
                    foreach ($categories as $category => $delete) {
                        if ($delete === true) {
                            $cat[$category] = true;
                        }
                    }

                    foreach (array_keys($cat) as $categoryId) {
                        $position = 1;
                        try {
                            $positions = $this-
>getCategoryPosition($categoryId);
                            $existingPositions = [];
                            foreach ($positions as $position) {
                                $existingPositions[] =
$position['position'];
                            }
                            if (!in_array(1, $existingPositions)) {
                                $position = 1;
                            } elseif ($maxPos =
max($existingPositions)) {
                                $position = $maxPos + 1;
                            }
                        } catch (Exception $exception) {
                            $position = 1;
                        }
                        $categoriesIn[] = [
                            'product_id' => $productId,
                            'category_id' => $categoryId,
                            'position' => $position
```

```
                        ];
                    }
                }
                if ($removeCategories || $config) {
                    $this->_connection->delete(
                        $tableName,
                        $this->_connection->quoteInto('product_id IN
(?)', $delProductId)
                    );
                }
                if ($categoriesIn) {
                    $this->_connection->insertOnDuplicate($tableName,
$categoriesIn, ['product_id', 'category_id']);
                }
            }
            return $this;
        }
        return parent::_saveProductCategories($categoriesData);
    }

    /**
     * @param array $categoryProductPosition
     * @return $this
     */
    protected function _saveProductCategoriesPosition(array
$categoryProductPosition)
    {
        static $tableName = null;

        if (!$tableName) {
            $tableName = $this->_resource-
>getTable('catalog_category_product');
        }

        $positionsIn = [];
        foreach ($categoryProductPosition as $sku => $categories) {
            $productId = $this->skuProcessor-
>getNewSku($sku)['entity_id'];
            foreach ($categories as $categoryId => $position) {
                $positionsIn[] = ['product_id' => $productId,
'category_id' => $categoryId, 'position' => $position];
            }
        }

        if ($positionsIn) {
            $this->_connection->insertOnDuplicate($tableName,
$positionsIn, ['position']);
        }

        return $this;
    }

    /**
     * Retrieve default attribute value (where store_id = 0)
     *
     * @param AbstractAttribute $attribute
     * @param string $sku
     * @param int $storeId
     * @return bool|string
     * @throws Exception
     */
    protected function getDefaultAttrValue(AbstractAttribute
$attribute, $sku, $storeId = 0)
    {
        if (!isset($this->_oldSku[strtolower($sku)])) {
            return false;
        }

        $linkField = $this->getProductEntityLinkField();
        $linkId = $this->_oldSku[strtolower($sku)][$linkField];

        $bind = [
            'attribute_id' => $attribute->getId(),
            'store_id' => $storeId,
            $linkField => $linkId,
        ];

        $select = $this->_connection->select()
            ->from($attribute->getBackend()->getTable(), 'value')
            ->where('attribute_id = :attribute_id')
            ->where('store_id = :store_id')
            ->where($linkField . ' = :' . $linkField);

        return $this->_connection->fetchOne($select, $bind);
    }

    /**
     * @param AbstractAttribute $attribute
     * @param string $sku
     * @param int $storeId
     * @return bool|string
     * @throws Exception
     */
    protected function _deleteStoreAttributeValue(AbstractAttribute
$attribute, $sku, $storeId = 0)
    {
        if (!isset($this->_oldSku[strtolower($sku)])) {
            return false;
        }

        $linkField = $this->getProductEntityLinkField();
        $linkId = $this->_oldSku[strtolower($sku)][$linkField];

        $deleteCondition[] = $this->_connection->quoteInto(
            'attribute_id = ?',
            $attribute->getId()
        );
        $deleteCondition[] = $this->_connection->quoteInto(
            'store_id = ?',
            $storeId
        );
        $deleteCondition[] = $this->_connection->quoteInto(
            $linkField . ' = ?',
            $linkId
        );

        $deleteCondition = implode(' AND ', $deleteCondition);

        return $this->_connection->delete($attribute->getBackend()-
>getTable(), $deleteCondition);
    }

    /**
     * Init media gallery resources.
     *
     * @return void
     */
    public function initMediaGalleryResources()
    {
        if (null == $this->mediaGalleryTableName) {
            $this->productEntityTableName = $this->getResource()-
>getTable('catalog_product_entity');
            $this->mediaGalleryTableName = $this->getResource()-
>getTable('catalog_product_entity_media_gallery');
            $this->mediaGalleryValueTableName = $this->getResource()-
>getTable(
                'catalog_product_entity_media_gallery_value'
            );
            $this->mediaGalleryEntityToValueTableName = $this-
>getResource()->getTable(

'catalog_product_entity_media_gallery_value_to_entity'
            );
        }
    }

    /**
     * @param $newMediaValues
     * @return $this
     */
    protected function removeExistingImages($newMediaValues)
    {
        $this->importImageProcessor-
>removeExistingImages($newMediaValues);
        return $this;
    }

    /**
     * @param string $sku
     * @return $this
     */
    protected function removeRelatedProducts($sku)
    {
        if (!isset($this->_oldSku[strtolower($sku)])) {
            return $this;
        }
        try {
            $entityLinkField = $this->getProductEntityLinkField();
            $productId = $this-
>_oldSku[strtolower($sku)][$entityLinkField];
            $linkTypeId = ProductLink::LINK_TYPE_RELATED;
            $linkTable = $this->getResource()-
>getTable('catalog_product_link');
            $this->_connection->delete(
                $linkTable,
                ['product_id=' . $productId, 'link_type_id=' .
$linkTypeId]
            );
        } catch (Exception $e) {
            $this->addLogWriteln($e->getMessage(), $this->output,
'error');
        }

        return $this;
    }

    /**
     * @param $sku
     *
     * @return $this
     */
    protected function removeCrosssellProducts($sku)
    {
        if (!isset($this->_oldSku[strtolower($sku)])) {
            return $this;
        }
        try {
            $entityLinkField = $this->getProductEntityLinkField();
            $productId = $this-
>_oldSku[strtolower($sku)][$entityLinkField];
            $linkTypeId = ProductLink::LINK_TYPE_CROSSSELL;
            $linkTable = $this->getResource()-
>getTable('catalog_product_link');
            $this->_connection->delete(
                $linkTable,
                ['product_id=' . $productId, 'link_type_id=' .
$linkTypeId]
            );
        } catch (Exception $e) {
```

```
            $this->addLogWriteln($e->getMessage(), $this->output,
'error');
        }

        return $this;
    }

    /**
     * @param $sku
     *
     * @return $this
     */
    protected function removeUpsellProducts($sku)
    {
        if (!isset($this->_oldSku[strtolower($sku)])) {
            return $this;
        }
        try {
            $entityLinkField = $this->getProductEntityLinkField();
            $productId = $this-
>_oldSku[strtolower($sku)][$entityLinkField];
            $linkTypeId = ProductLink::LINK_TYPE_UPSELL;
            $linkTable = $this->getResource()-
>getTable('catalog_product_link');
            $this->_connection->delete(
                $linkTable,
                ['product_id=' . $productId, 'link_type_id=' .
$linkTypeId]
            );
        } catch (Exception $e) {
            $this->addLogWriteln($e->getMessage(), $this->output,
'error');
        }

        return $this;
    }

    /**
     * Initialize source type model
     *
     * @param $type
     *
     * @throws LocalizedException
     */
    protected function _initSourceType($type)
    {
        if (!$this->sourceType) {
            $this->sourceType = $this->additional-
>getSourceModelByType($type);
            $this->sourceType->setData($this->_parameters);
        }
    }

    /**
     * Import images via initialized source type
     *
     * @param $bunch
     *
     * @return mixed
     */
    protected function prepareImagesFromSource($bunch)
    {
        foreach ($bunch as $rowNum => &$rowData) {
            $rowData = $this->customFieldsMapping($rowData);
            foreach ($this->_imagesArrayKeys as $image) {
                if (empty($rowData[$image])) {
                    continue;
                }
                $dispersionPath =
\Magento\Framework\File\Uploader::getDispretionPath($rowData[$image]);
                $importImages = explode($this-
>getMultipleValueSeparator(), $rowData[$image]);
                $imageArr = [];
                foreach ($importImages as $importImage) {
                    $imageSting = mb_strtolower(
                        $dispersionPath . '/' . preg_replace('/[^a-
z0-9\._-]+/i', '', $importImage)
                    );
                    if ($this->sourceType) {
                        if ($this->sourceType->getCode() === 'rest')
{
                            $sourceImport = $this->sourceType-
>importImage($importImage, $imageSting);
                            $imageArr[] = $this->sourceType-
>getCode() . $sourceImport[1];
                        } else {
                            $this->sourceType-
>importImage($importImage, $imageSting);
                        }
                    }
                    if ($this->sourceType->getCode() !== 'rest') {
                        $imageArr[] = $this->sourceType->getCode() .
$imageSting;
                    }
                }
                $rowData[$image] = implode($this-
>getMultipleValueSeparator(), $imageArr);
            }
        }

        return $bunch;
    }

    /**
     * @param array $rowData
     * @param null $storeIds
     * @return array
     * @throws Exception
     */
    protected function generateUrlKey(array $rowData, $storeIds =
null)
    {
        $productEntityLinkField = $this->getProductEntityLinkField();
        $sku = $this->getCorrectSkuAsPerLength($rowData);
        $urlKey = $rowData[self::URL_KEY] ?? '';
        $name = $rowData[self::COL_NAME] ?? '';
        if ($this->isSkuExist($sku)) {
            $exiting = $this->getExistingSku($sku);
            if (!$urlKey) {
                $attr = $this-
>retrieveAttributeByCode(self::URL_KEY);
                $select = $this->getConnection()->select()
                    ->from($attr->getBackendTable(), ['value'])
                    ->where($productEntityLinkField . ' = (?)',
$exiting['entity_id'])
                    ->where('attribute_id = (?)', $attr-
>getAttributeId());
                $urlKey = $this->getConnection()->fetchOne($select);
            }
            if (!$name) {
                $attr = $this-
>retrieveAttributeByCode(self::COL_NAME);
                $select = $this->getConnection()->select()
                    ->from($attr->getBackendTable(), ['value'])
                    ->where($productEntityLinkField . ' = (?)',
$exiting['entity_id'])
                    ->where('attribute_id = (?)', $attr-
>getAttributeId());
                $name = $this->getConnection()->fetchOne($select);
                if (!$urlKey) {
                    $urlKey = $name;
                }
            }
        } else {
            $urlKey = isset($rowData[self::URL_KEY])
                ? $urlKey
                : $name;
        }
        if ($storeIds === null) {
            $storeIds = $this->getStoreIds();
        }
        $urlKey = ($urlKey != '') ?
            $this->productUrl->formatUrlKey($urlKey)
            : $this->productUrl->formatUrlKey($name);
        $isDuplicate = $this->isDuplicateUrlKey($urlKey, $sku,
$storeIds);
        if ($isDuplicate || $this->urlKeyManager->isUrlKeyExist($sku,
$urlKey)) {
            $urlKey = $this->productUrl->formatUrlKey(
                $name . '-' . $sku
            );
        }
        $rowData[self::URL_KEY] = $urlKey;
        $this->urlKeyManager->addUrlKeys($sku, $urlKey);
        return $rowData;
    }

    /**
     * Custom fields mapping for changed purposes of fields and field
names.
     *
     * @param array $rowData
     *
     * @return array
     */
    public function customFieldsMapping($rowData)
    {
        $rowData = $this->attributeValuesMapping($rowData);

        foreach ($this->_fieldsMap as $systemFieldName =>
$fileFieldName) {
            if (array_key_exists($fileFieldName, $rowData) &&
!isset($rowData[$systemFieldName])) {
                $rowData[$systemFieldName] =
$rowData[$fileFieldName];
            }
        }
        // restore data for configurable field when it is already
used in Map Attributes section
        $configField = $this->_parameters['configurable_field'];
        if ($configField && !isset($rowData[$configField])) {
            $configKey = array_search($configField, $this-
>_fieldsMap);
            if ($configKey !== false) {
                $rowData[$configField] = $rowData[$configKey];
            }
        }
        //
        $rowData = $this->_parseAdditionalAttributes($rowData);
        $rowData = $this->setStockUseConfigFieldsValues($rowData);

        if (array_key_exists('status', $rowData)
            && $rowData['status'] != Status::STATUS_ENABLED
        ) {
            if ($rowData['status'] == 'yes') {
                $rowData['status'] = Status::STATUS_ENABLED;
            } elseif (!empty($rowData['status']) || $this-
>getRowScope($rowData) == self::SCOPE_DEFAULT) {
                $rowData['status'] = Status::STATUS_DISABLED;
            }
```

```
        }
        $imageImportPath = $this-
>_parameters[Import::FIELD_NAME_IMG_FILE_DIR] ?? '';
        if (preg_match('/\bhttps?:\/\//i', $imageImportPath,
$matches)) {
            foreach ($this->_imagesArrayKeys as $image) {
                if (empty($rowData[$image]) &&
!isset($rowData[$image])) {
                    continue;
                }
                if ($image === 'additional_images') {
                    $addImage = [];
                    foreach (explode($this-
>getMultipleValueSeparator(), $rowData[$image]) as $rowImage) {
                        $addImage[] = $imageImportPath . '/' .
$rowImage;
                    }
                    $rowData[$image] = implode($this-
>getMultipleValueSeparator(), $addImage);
                } else {
                    $rowData[$image] = $imageImportPath . '/' .
$rowData[$image];
                }
            }
        }

        foreach ($this->_imagesArrayKeys as $image) {
            if ($image != '_media_image') {
                if (isset($rowData[$image])) {
                    $rowData[$image] = trim($rowData[$image]);
                }
            }
        }

        return $rowData;
    }

    /**
     * Parse attributes names and values string to array.
     *
     * @param array $rowData
     *
     * @return array
     */
    private function _parseAdditionalAttributes($rowData)
    {
        if (empty($rowData['additional_attributes'])) {
            return $rowData;
        }
        try {
            $source = $this->_getSource();
        } catch (Exception $e) {
            $source = null;
        }
        $valuePairs = explode(
            $this->getMultipleValueSeparator(),
            $rowData['additional_attributes']
        );
        foreach ($valuePairs as $valuePair) {
            $separatorPosition = strpos($valuePair,
self::PAIR_NAME_VALUE_SEPARATOR);
            if ($separatorPosition !== false) {
                $key = substr($valuePair, 0, $separatorPosition);
                $value = substr(
                    $valuePair,
                    $separatorPosition +
strlen(self::PAIR_NAME_VALUE_SEPARATOR)
                );
                if ($source !== null) {
                    $key = $source->changeField($key);
                }
                $multiLineSeparator = strpos($value,
self::PSEUDO_MULTI_LINE_SEPARATOR);
                if ($multiLineSeparator !== false) {
                    $attribute = $this-
>retrieveAttributeByCode($key);
                    if ($attribute
                        && $attribute->getBackendType() !== 'varchar'
                        && $attribute->getBackendType() !== 'text'
                        && !$this->verifySwatchString($value)
                    ) {
                        $value = implode(
                            $this->getMultipleValueSeparator(),
                            explode(
                                self::PSEUDO_MULTI_LINE_SEPARATOR,
                                $value
                            )
                        );
                    }
                }
                $rowData[$key] = $value === false ? '' : $value;
            }
        }

        return $rowData;
    }

    /**
     * @param string $value
     * @return bool
     */
    private function verifySwatchString(string $value)
    {
        return (strpos($value, 'type=') !== false && strpos($value,
'value=') !== false)
            ? true
            : false;
    }

    /**
     * Set values in use_config_ fields.
     *
     * @param array $rowData
     *
     * @return array
     */
    private function setStockUseConfigFieldsValues($rowData)
    {
        $useConfigFields = [];
        foreach ($rowData as $key => $value) {
            if (isset($this->defaultStockData[$key])
                && isset($this-
>defaultStockData[self::INVENTORY_USE_CONFIG_PREFIX . $key])
                && !empty($value)
            ) {
                $useConfigFields[self::INVENTORY_USE_CONFIG_PREFIX .
$key] =
                    ($value == self::INVENTORY_USE_CONFIG) ? 1 : 0;
            }
        }
        if (!empty($useConfigFields)) {
            $rowData = array_merge($rowData, $useConfigFields);
        }

        return $rowData;
    }

    /**
     * Replace attribute values according map
     *
     * @param array $rowData
     *
     * @return array
     */
    protected function attributeValuesMapping($rowData)
    {
        /** @var
\Ivan\ImportExport\Model\ResourceModel\Job\Collection $collection */
        $collection = $this->importCollection->create();
        $collection->addFieldToFilter($collection->getIdFieldName(),
$this->_parameters['job_id']);

        /** @var Job $job */
        foreach ($collection as $job) {
            $map = Serializer::unserialize($job->getMapping());

            foreach ($map as $item) {
                if
(isset($item['source_data_attribute_value_system']) &&

isset($item['source_data_attribute_value_import'])
                ) {
                    foreach ($rowData as $key => $value) {
                        if (!is_array($value) && (trim($value) ==
trim($item['source_data_attribute_value_import']))) {
                            $rowData[$key] =
$item['source_data_attribute_value_system'];
                        }
                    }
                }
            }
        }

        return $rowData;
    }

    /**
     * @param $rowData
     * @return array
     */
    protected function categoriesMapping($rowData)
    {
        $categories_separator = $this-
>_parameters['categories_separator'];
        $explodeImportedCategoriesItems =
explode($categories_separator, $rowData[self::COL_CATEGORY]);

        $explodeImportedCategoriesPositionItems = [];
        $importedCategoriesPositionItems =
(!empty($rowData[self::COL_CATEGORY . '_position']))
            ? explode($categories_separator,
$rowData[self::COL_CATEGORY . '_position'])
            : [];
        foreach ($importedCategoriesPositionItems as $index =>
$positionItem) {
            if (empty($positionItem)) {
                continue;
            }
            $categoriesPositionValue =
explode(self::PAIR_NAME_VALUE_SEPARATOR, $positionItem);

$explodeImportedCategoriesPositionItems[$categoriesPositionValue[0]] =
[
                $index,
                $categoriesPositionValue[1]
            ];
        }

        $connection = $this->_connection;
        $resource = $this->getResource();
```

```
        $select = $connection->select()->from(
            [
                'main' => $resource->getTable('Ivan_import_jobs'),
            ],
            ['mapping']
        )->where('entity_id=?', $this->_parameters['job_id']);
        $maps = $this->_connection->fetchAll(
            $select
        );
        foreach ($maps as $map) {
            $newCategoriesMapItems =
Serializer::unserialize($map['mapping']);
            foreach ($newCategoriesMapItems as $newCategoriesMapItem)
{
                foreach ($explodeImportedCategoriesItems as
&$explodeImportedCategoriesItem) {
                    if
(isset($newCategoriesMapItem['source_category_data_import']) &&
                        trim($explodeImportedCategoriesItem) ==
$newCategoriesMapItem['source_category_data_import']
                    ) {
                        $explodeImportedCategoriesItem =
$newCategoriesMapItem['source_category_data_new'];
                        $this->setIsRowCategoryMapped(true);

                        if
(isset($explodeImportedCategoriesPositionItems[$explodeImportedCategor
iesItem])) {
                            $index =
$explodeImportedCategoriesPositionItems[$explodeImportedCategoriesItem
][0];
                            $position =
$explodeImportedCategoriesPositionItems[$explodeImportedCategoriesItem
][1];
                            $importedCategoriesPositionItems[$index]
= implode(
                                self::PAIR_NAME_VALUE_SEPARATOR,

[$newCategoriesMapItem['source_category_data_new'], $position]
                            );
                        }
                    }
                }
            }
        }
        return [
            self::COL_CATEGORY => implode($categories_separator,
$explodeImportedCategoriesItems),
            self::COL_CATEGORY . '_position' =>
implode($categories_separator, $importedCategoriesPositionItems)
        ];
    }

    /**
     * @param MagentoProduct\Type\AbstractType $productTypeModel
     * @param array $rowData
     * @return array
     * @throws LocalizedException
     */
    public function createAttributeValues(
        MagentoProduct\Type\AbstractType $productTypeModel,
        array $rowData
    ) {
        $options = [];
        if (isset($rowData[self::COL_ATTR_SET])) {
            $attributeSet = $rowData[self::COL_ATTR_SET];
            foreach ($rowData as $attrCode => $attrValue) {
                /**
                 * Add attribute to set & set's group
                 */
                if (preg_match('/^(attribute\|).+/', $attrCode)) {
                    $columnData = explode('|', $attrCode);
                    $columnData = $this-
>prepareAttributeData($columnData);
                    // might be already inside additional_attributes
                    if
(isset($rowData[$columnData['attribute_code']])) {
                        unset($rowData[$attrCode]);
                        continue;
                    } else {
                        $rowData[$columnData['attribute_code']] =
$rowData[$attrCode];
                        unset($rowData[$attrCode]);
                        $attrCode = $columnData['attribute_code'];
                    }
                }

                /**
                 * Prepare new values
                 */
                $attrParams = $productTypeModel-
>retrieveAttribute($attrCode, $attributeSet);
                if (!empty($attrParams)) {
                    if (!$attrParams['is_static'] &&
isset($rowData[$attrCode]) && trim($rowData[$attrCode]) !== '') {
                        switch ($attrParams['type']) {
                            case 'select':
                                $swatchOptionData = [];
                                $swatchOptions = [];
                                /** @var
\Magento\Catalog\Model\ResourceModel\Eav\Attribute\Interceptor
$attribute */
                                $attribute = $this-
>retrieveAttributeByCode($attrCode);
                                if ($this->swatchesHelperData-
>isVisualSwatch($attribute)) {
                                    $swatchOptionData = $this-
>prepareSwatchOptionData($rowData[$attrCode]);
                                    $swatchOptions = $this-
>getSwatchesByOptionsId(
                                        $attrParams['options'],
                                        $attrParams['id']
                                    );
                                }
                                if ($this->swatchesHelperData-
>isTextSwatch($attribute)) {
                                    $swatchOptionData =
$rowData[$attrCode];
                                    $swatchOptions = $this-
>getSwatchesByOptionsId(
                                        $attrParams['options'],
                                        $attrParams['id']
                                    );
                                }
                                if
((isset($attrParams['additional_data']['update_product_preview_image']
)
                                    ||
isset($attrParams['additional_data']['use_product_image_for_swatch']))
                                    && $this-
>verifySwatchString($rowData[$attrCode])
                                ) {
                                    $swatchOptionData =
$rowData[$attrCode];
                                    $swatchOptions = $this-
>getSwatchesByOptionsId(
                                        $attrParams['options'],
                                        $attrParams['id']
                                    );
                                }
                                $lowerAttrValue =
strtolower($rowData[$attrCode]);
                                // no attribute option
                                $storeCode =
isset($rowData[self::COL_STORE_VIEW_CODE])
                                    ?
$rowData[self::COL_STORE_VIEW_CODE] : false;
                                $scopeStore = self::SCOPE_STORE ==
$this->getRowScope($rowData);
                                $attrOptions =
$attrParams['options'];
                                if ($scopeStore && $storeCode &&
!empty($attrParams['options_store'][$storeCode])) {
                                    if
(isset($attrOptions[$lowerAttrValue]) &&

!isset($attrParams['options_store'][$storeCode][$lowerAttrValue])
                                    ) {

$attrParams['options_store'][$storeCode][$lowerAttrValue] =

$attrOptions[$lowerAttrValue];
                                    }
                                    $attrOptions =
$attrParams['options_store'][$storeCode];
                                }

                                if
(!isset($attrOptions[$lowerAttrValue])) {
                                    $options[$attrParams['id']][] = [
                                        'sort_order' =>
count($attrParams['options']) + 1,
                                        'value' =>
$rowData[$attrCode],
                                        'code' => $attrCode,
                                        'swatch_option' =>
$swatchOptionData,
                                    ];
                                } elseif (!empty($swatchOptionData)
&&

!array_key_exists($attrOptions[$lowerAttrValue], $swatchOptions)
                                ) { // no attribute swatch option

$newSwatchOptions[$attrParams['id']][$attrOptions[$lowerAttrValue]] =
                                        $swatchOptionData;
                                } elseif
(array_key_exists($attrOptions[$lowerAttrValue], $swatchOptions)) {
                                    // swatch attribute option exist
                                    $swatchOption =
$swatchOptions[$attrOptions[$lowerAttrValue]];
                                    if ($this->swatchesHelperData-
>isVisualSwatch($attribute)) {
                                        // but has different value or
type
                                        if
(isset($attrParams['additional_data']['update_product_preview_image'])
                                            ||
isset($attrParams['additional_data']['use_product_image_for_swatch'])
                                            &&
!is_array($swatchOptionData)
                                        ) {
                                            break; //continue 2
                                        }
                                        if (!empty($diff =
array_diff_assoc($swatchOptionData, $swatchOption))) {
                                            if
((array_key_exists('type', $diff) &&
```

```
                                                $diff['type'] ==
Swatch::SWATCH_TYPE_VISUAL_COLOR)
                                            ||
(!array_key_exists('type', $diff) &&

$swatchOption['type'] == Swatch::SWATCH_TYPE_VISUAL_COLOR)
                                        ) {
                                            $this-
>updateSwatchOption($swatchOption, $diff);
                                        } elseif ($this-
>ifVisualSwatchOptionDifferent($swatchOption, $diff)) {
                                            $diff['value'] =
$this->uploadVisualSwatchFile($diff['value']);
                                            $this-
>updateSwatchOption($swatchOption, $diff);
                                        }
                                    }
                                }
                            }
                            break;
                        case 'multiselect':
                            $separator = $this-
>_parameters['_import_multiple_value_separator'] ?
                                $this-
>_parameters['_import_multiple_value_separator'] :

MagentoProduct::PSEUDO_MULTI_LINE_SEPARATOR;
                            $values = explode($separator,
$rowData[$attrCode]);
                            foreach ($values as $value) {
                                $value = trim($value);
                                if
(!isset($attrParams['options'][strtolower($value)])) {
                                    $options[$attrParams['id']][]
= [
                                        'sort_order' =>
count($attrParams['options']) + 1,
                                        'value' => $value,
                                        'code' => $attrCode,
                                    ];
                                }
                            }
                            break;
                        default:
                            break;
                    }
                }
            }
        }

        /**
         * Create new values
         */
        if (!empty($options)) {
            $connection = $this->_connection;
            $resource = $this->getResource();
            foreach ($options as $attributeId => $optionsArray) {
                foreach ($optionsArray as $option) {
                    /**
                     * @see
\Magento\Eav\Model\ResourceModel\Entity\Attribute::_updateAttributeOpt
ion()
                     */
                    $table = $resource-
>getTable('eav_attribute_option');
                    $data = ['attribute_id' => $attributeId,
'sort_order' => $option['sort_order']];
                    $connection->insert($table, $data);
                    $intOptionId = $connection-
>lastInsertId($table);
                    /**
                     * @see
\Magento\Eav\Model\ResourceModel\Entity\Attribute::_updateAttributeOpt
ionValues()
                     */
                    $table = $resource-
>getTable('eav_attribute_option_value');
                    $data = ['option_id' => $intOptionId,
'store_id' => 0, 'value' => $option['value']];
                    $connection->insert($table, $data);
                    if (isset($option['swatch_option']) &&
!empty($option['swatch_option'])) {
                        $this->insertNewSwatchOption(
                            $connection,
                            $resource,
                            $intOptionId,
                            $option['swatch_option'],
                            $attributeId
                        );
                    }
                    foreach ($this->_productTypeModels as
$productTypeModel) {
                        $productTypeModel->addAttributeOption(
                            $option['code'],
                            strtolower($option['value']),
                            $intOptionId
                        );
                    }
                }
            }
        }
        if (!empty($newSwatchOptions)) {
            $connection = $this->_connection;
            $resource = $this->getResource();
            foreach ($newSwatchOptions as $attributeId =>
$swatchOption) {
                foreach ($swatchOption as $optionId =>
$swatchData) {
                    $this->insertNewSwatchOption($connection,
$resource, $optionId, $swatchData, $attributeId);
                }
            }
        }

        return $rowData;
    }

    /**
     * Convert attribute string syntax to array.
     *
     * @param $columnData
     *
     * @return array
     * @throws Exception
     */
    protected function prepareAttributeData($columnData)
    {
        $result = [];
        foreach ($columnData as $field) {
            $field = explode(':', $field);
            if (isset($field[1])) {
                if (preg_match('/^(frontend_label_)[0-9]+/',
$field[0])) {
                    $result['frontend_label'][(int)substr($field[0],
-1)] = $field[1];
                } else {
                    $result[$field[0]] = $field[1];
                }
            }
        }

        if (!empty($result)) {
            $attributeCode = isset($result['attribute_code']) ?
$result['attribute_code'] : null;
            $frontendLabel = $result['frontend_label'][0];
            $attributeCode = $attributeCode ?: $this-
>generateAttributeCode($frontendLabel);
            $result['attribute_code'] = $attributeCode;

            $entityTypeId = $this->eavEntityFactory->create()-
>setType(
                \Magento\Catalog\Model\Product::ENTITY
            )->getTypeId();
            $result['entity_type_id'] = $entityTypeId;
            $result['is_user_defined'] = 1;
        }

        return $result;
    }

    /**
     * Generate code from label
     *
     * @param string $label
     * @return string
     * @throws Zend_Validate_Exception
     */
    protected function generateAttributeCode($label)
    {
        $code = substr(
            preg_replace(
                '/[^a-z_0-9]/',
                '_',
                $this->productUrl->formatUrlKey($label)
            ),
            0,
            30
        );
        $validatorAttrCode = new Zend_Validate_Regex(['pattern' =>
'/^[a-z][a-z_0-9]{0,29}[a-z0-9]$/']);
        if (!$validatorAttrCode->isValid($code)) {
            $code = 'attr_' . ($code ?: substr(hash("md5", time()),
0, 8));
        }

        return $code;
    }

    /**
     * Parse swatch attribute value, pulls actual attribute value and
swatch options if they are there.
     *
     * @param $attributeValue string Swatch attribute value in follow
format
     *     "{value}|type={1,2}|value={#FFFFFF,path/to/image.file}"
where: type:1 - color type:2 - image
     *
     * @return array
     */
    protected function prepareSwatchOptionData(&$attributeValue)
    {
        $swatchOptionData = [];
        $preParsedData = explode('|', $attributeValue);
        if (count($preParsedData) > 1) {
            foreach ($preParsedData as $key => $value) {
                if ($key == 0) {
                    $attributeValue = $value; //set attributes value
                    continue;
```

```
                }
                $value = explode('=', $value);
                if (isset($value[1])) {
                    $swatchOptionData[$value[0]] = $value[1];
                }
            }
        }

        return $swatchOptionData;
    }

    /**
     * Returns Swatch option data for Attribute Option Ids
     *
     * @param array $optionIds
     * @param int $attributeId
     *
     * @return array
     */
    protected function getSwatchesByOptionsId($optionIds,
$attributeId)
    {
        if (!isset($this->cachedSwatchOptions[$attributeId]) ||
empty($this->cachedSwatchOptions[$attributeId])) {
            $this->cachedSwatchOptions[$attributeId] = [];
            $swatchCollection = $this->swatchCollectionFactory-
>create();
            $swatchCollection->addFilterByOptionsIds($optionIds);
            foreach ($swatchCollection as $item) {
                $this-
>cachedSwatchOptions[$attributeId][$item['option_id']] = $item-
>getData();
            }
        }

        return $this->cachedSwatchOptions[$attributeId];
    }

    /**
     * @param int $swatchOption
     * @param array $diff
     */
    protected function updateSwatchOption($swatchOption, $diff)
    {
        $connection = $this->_connection;
        $resource = $this->getResource();
        $table = $resource->getTable('eav_attribute_option_swatch');
        if (isset($swatchOption['swatch_id'])) {
            $where = ['swatch_id=?' =>
(int)$swatchOption['swatch_id']];
            $connection->update($table, $diff, $where);
        }
    }

    /**
     * Checks if imported image for swatch option is different then
exist one.
     *
     * @param int $swatchOption
     * @param array $diff Array of type and value that are different
     *
     * @return bool
     * @throws LocalizedException
     */
    protected function ifVisualSwatchOptionDifferent($swatchOption,
$diff)
    {
        // TODO: need implement logic for unique names -
        // sometimes image name might have _1_2 endings for the same
image.
        if (isset($diff['value'])) {
            $fileName = preg_replace('/[^a-z0-9\._-]+/i', '',
$diff['value']);
            $dispretionPath = $this->_getUploader()-
>getDispretionPath($fileName);
            return !($swatchOption['value'] == $dispretionPath . '/'
. $fileName);
        }
        return false;
    }

    /**
     * @return
\Magento\CatalogImportExport\Model\Import\Uploader|Uploader
     * @throws LocalizedException
     */
    protected function _getUploader()
    {
        $DS = DIRECTORY_SEPARATOR;
        if ($this->_fileUploader === null) {
            $this->_fileUploader = $this->_uploaderFactory->create();
            $this->_fileUploader->init();
            $dirConfig = DirectoryList::getDefaultConfig();
            $dirAddon =
$dirConfig[DirectoryList::MEDIA][DirectoryList::PATH];
            if (!empty($this-
>_parameters[Import::FIELD_NAME_IMG_FILE_DIR])) {
                $tmpPath = $this-
>_parameters[Import::FIELD_NAME_IMG_FILE_DIR];
            } else {
                $tmpPath = $dirAddon . $DS . $this->_mediaDirectory-
>getRelativePath('import');
            }
            if (preg_match('/\bhttps?:\/\//i', $tmpPath, $matches)) {
                $tmpPath = $dirAddon . $DS . $this->_mediaDirectory-
>getRelativePath('import');
            }
            if (!$this->_fileUploader->setTmpDir($tmpPath)) {
                $this->addLogWriteln(__('File directory \'%1\' is not
readable.', $tmpPath), $this->output, 'info');
                $this->addRowError(
                    __('File directory \'%1\' is not readable.',
$tmpPath),
                    null,
                    null,
                    null,
                    ProcessingError::ERROR_LEVEL_NOT_CRITICAL
                );
                throw new LocalizedException(
                    __('File directory \'%1\' is not readable.',
$tmpPath)
                );
            }
            $destinationDir = "catalog/product";
            $destinationPath = $dirAddon . $DS . $this-
>_mediaDirectory->getRelativePath($destinationDir);

            $this->_mediaDirectory->create($destinationPath);
            if (!$this->_fileUploader->setDestDir($destinationPath))
{
                $this->addRowError(
                    __('File directory \'%1\' is not writable.',
$destinationPath),
                    null,
                    null,
                    null,
                    ProcessingError::ERROR_LEVEL_NOT_CRITICAL
                );
                throw new LocalizedException(
                    __('File directory \'%1\' is not writable.',
$destinationPath)
                );
            }
        }

        if ($this->_fileUploader) {
            $this->_fileUploader->setEntity($this);
            $this->_fileUploader->setOutput($this->getOutput());
        }

        return $this->_fileUploader;
    }

    /**
     * Uploads Image for Image Swatch option
     *
     * @param string $swatchVisualFile
     * @return string
     * @throws LocalizedException
     */
    protected function uploadVisualSwatchFile($swatchVisualFile)
    {
        $config = $this->mediaConfig;
        $uploader = $this->_getUploader();
        $newFile = '';
        $dirConfig = DirectoryList::getDefaultConfig();
        $mediaRelativePath =
$dirConfig[DirectoryList::MEDIA][DirectoryList::PATH];
        try {
            $destDir = $uploader->getDestDir();
            $uploadDir = $mediaRelativePath . DIRECTORY_SEPARATOR .
$config->getBaseTmpMediaPath();
            $uploadDir = $this->_mediaDirectory-
>getAbsolutePath($uploadDir);

            if (!$uploader->isDirectoryWritable($uploadDir)) {
                $uploader->createDirectory($uploadDir);
            }
            if (!$uploader->setDestDir($uploadDir)) {
                $this->addRowError(
                    __('File directory \'%1\' is not writable.',
$config->getBaseTmpMediaPath()),
                    null,
                    null,
                    null,
                    ProcessingError::ERROR_LEVEL_NOT_CRITICAL
                );
                throw new LocalizedException(
                    __('File directory \'%1\' is not writable.',
$config->getBaseTmpMediaPath())
                );
            } else {
                $result = $uploader->move($swatchVisualFile);
                $newFile = $this->swatchHelperMedia-
>moveImageFromTmp($result['file']);
                $this->swatchHelperMedia-
>generateSwatchVariations($newFile);
                $uploader->setDestDir($destDir);
            }
        } catch (Exception $e) {
            $this->addLogWriteln($e->getMessage(), $this->output,
'error');
        }

        return $newFile;
    }

    /**
     * @param AdapterInterface $connection
```

```
     * @param ResourceModel $resource
     * @param int $optionId
     * @param array $swatchData
     * @param int $attributeId
     * @return $this
     * @throws LocalizedException
     */
    protected function insertNewSwatchOption($connection, $resource,
$optionId, $swatchData, $attributeId)
    {
        if (!isset($swatchData['type']) &&
!isset($swatchData['value'])) {
            $table = $resource-
>getTable('eav_attribute_option_swatch');
            $data = [
                'option_id' => $optionId,
                'store_id' => 0,
                'type' => Swatch::SWATCH_TYPE_TEXTUAL,
                'value' => $swatchData,
            ];
            $connection->insert($table, $data);
            $this->cachedSwatchOptions[$attributeId][$optionId] =
$data;
        } else {
            if ($swatchData['type'] ==
Swatch::SWATCH_TYPE_VISUAL_IMAGE) {
                $swatchData['value'] = $this-
>uploadVisualSwatchFile($swatchData['value']);
            }
            if ($swatchData['value']) {
                $table = $resource-
>getTable('eav_attribute_option_swatch');
                $data = [
                    'option_id' => $optionId,
                    'store_id' => 0,
                    'type' => $swatchData['type'],
                    'value' => $swatchData['value'],
                ];
                $connection->insert($table, $data);
                $this->cachedSwatchOptions[$attributeId][$optionId] =
$data;
            }
        }

        return $this;
    }

    /**
     * @param $urlKey
     * @param $sku
     * @param $storeId
     *
     * @return string
     */
    protected function isDuplicateUrlKey($urlKey, $sku, $storeId)
    {
        $result = false;
        $urlKeyHtml = $urlKey . $this->getProductUrlSuffix();
        $resource = $this->getResource();
        $select = $this->_connection->select()->from(
            ['url_rewrite' => $resource->getTable('url_rewrite')],
            ['request_path', 'store_id']
        )->joinLeft(
            ['cpe' => $resource->getTable('catalog_product_entity')],
            'cpe.entity_id = url_rewrite.entity_id'
        )->where("request_path='$urlKey' OR
request_path='$urlKeyHtml'")
            ->where('store_id IN (?)', $storeId)
            ->where('cpe.sku not in (?)', $sku);
        $isDuplicate = $this->_connection->fetchAssoc(
            $select
        );
        if (!empty($isDuplicate)) {
            $result = true;
        }
        return $result;
    }

    /**
     * @param array $rowData
     * @return string
     * @throws Exception
     */
    protected function getUrlKey($rowData)
    {
        $url = $this->productUrl-
>formatUrlKey(parent::getUrlKey($rowData));
        $this->urlKeyManager->addUrlKeys($rowData[self::COL_SKU],
$url);
        return $url;
    }

    /**
     * Divide additionalImages for old Magento version
     * @param $rowData
     *
     * @return mixed
     */
    protected function checkAdditionalImages($rowData)
    {
        return $this->importImageProcessor-
>checkAdditionalImages($rowData);
    }

    /**
     * @param array $rowData
     * @return array
     */
    public function stripSlashes(array $rowData)
    {
        foreach ($rowData as $key => $val) {
            if ($key === '') {
                continue;
            }
            if (!empty($val)) {
                $rowData[$key] = stripslashes((string) $val);
            }
        }

        return $rowData;
    }

    /**
     * @param array $rowData
     *
     * @return array
     */
    public function prepareRowForDb(array $rowData)
    {
        $rowData = $this->customFieldsMapping($rowData);

        $this->stripSlashes($rowData);

        static $lastSku = null;

        if (Import::BEHAVIOR_DELETE == $this->getBehavior()) {
            return $rowData;
        }

        $lastSku = $this->getCorrectSkuAsPerLength($rowData);

        if (version_compare($this->productMetadata->getVersion(),
'2.2.0', '>=')) {
            $checkSku = strtolower($lastSku);
        } else {
            $checkSku = $lastSku;
        }
        if (isset($this->_oldSku[$checkSku]) && $this-
>_oldSku[$checkSku]) {
            $newSku = $this->skuProcessor->getNewSku($lastSku);
            if (isset($rowData[self::COL_ATTR_SET]) &&
!$rowData[self::COL_ATTR_SET]) {
                $rowData[self::COL_ATTR_SET] =
$newSku['attr_set_code'];
            }
            if (isset($rowData[self::COL_TYPE]) &&
!$rowData[self::COL_TYPE]) {
                $rowData[self::COL_TYPE] = $newSku['type_id'];
            }
        }

        return $rowData;
    }

    /**
     * @param $rowData
     *
     * @return mixed
     */
    public function applyCategoryLevelSeparator($rowData)
    {
        $defaultCategoryName = '';
        $importCategoryName = '';
        if (isset($this->_parameters['root_category_id']) && $this-
>_parameters['root_category_id'] > 0) {
            $importCategoryId = (int)$this-
>_parameters['root_category_id'];
            /** @var \Magento\Catalog\Model\Category $importCategory
*/
            $importCategory = $this->categoryProcessor-
>getCategoryById($importCategoryId);
            if ($importCategory) {
                if ($importCategory->getParentCategory()->getId() !=
1) {
                    $importCategoryName = $defaultCategoryName =
$importCategory->getParentCategory()->getName();
                }
                if ((int)$importCategory->getId() ===
$importCategoryId && $importCategoryName !== '') {
                    $importCategoryName .= '/' . $importCategory-
>getName();
                } else {
                    $defaultCategoryName = $importCategoryName =
$importCategory->getName();
                }
            }
        } elseif (isset($rowData['_root_category'])) {
            $importCategoryName = $defaultCategoryName =
$rowData['_root_category'];
        }
        if (isset($rowData[self::COL_CATEGORY]) &&
$rowData[self::COL_CATEGORY]) {
            if ($this->_parameters['category_levels_separator'] !==
'/') {
                $rowData[self::COL_CATEGORY] = str_replace("/", "\/",
$rowData[self::COL_CATEGORY]);
            }
            $rowData[self::COL_CATEGORY] = str_replace(
                $this->_parameters['category_levels_separator'],
                '/',
```

```
                $rowData[self::COL_CATEGORY]
            );

            $rowCategories = explode('/',
$rowData[self::COL_CATEGORY]);
            $finalRowCat = [];
            foreach ($rowCategories as $rowCat) {
                if ($rowCat == '') {
                    continue;
                }
                $finalRowCat[] = $rowCat;
            }
            $rowData[self::COL_CATEGORY] = implode('/',
$finalRowCat);
        }
        $categories = [];
        if ($defaultCategoryName && $importCategoryName &&
isset($rowData[self::COL_CATEGORY])) {
            foreach (explode($this-
>_parameters['categories_separator'], $rowData[self::COL_CATEGORY]) as
$category) {
                if (strpos(trim($category), $defaultCategoryName) !==
false) {
                    $categories[] = trim($category);
                } else {
                    $categories[] = $importCategoryName . '/' .
trim($category);
                }
            }
            $rowData[self::COL_CATEGORY] = implode($this-
>_parameters['categories_separator'], $categories);
        } elseif ($importCategoryName) {
            $rowData[self::COL_CATEGORY] = $importCategoryName;
        }
        return $rowData;
    }

    /**
     * @param $array
     *
     * @return array
     */
    protected function deleteEmpty($array)
    {
        if (isset($array[self::COL_SKU])) {
            unset($array[self::COL_SKU]);
        }
        $newElement = [];
        foreach ($array as $key => $element) {
            if (strlen($element)) {
                $newElement[$key] = $element;
            }
        }

        return $newElement;
    }

    protected function getCategories($rowData)
    {
        if (isset($rowData[self::COL_STORE])) {
            $this->categoryProcessor->setStoreId($this-
>storeResolver->getStoreCodeToId($rowData[self::COL_STORE]));
        }
        $this->categoryProcessor->setGeneratUrl($this-
>_parameters['generate_url']);
        $this->categoryProcessor->setResource($this->getResource());
        $ids = $this->categoryProcessor->getRowCategories($rowData,
$this->_parameters['categories_separator']);
        foreach ($this->categoryProcessor->getFailedCategories() as
$error) {
            $this->errorAggregator->addError(
                AbstractEntity::ERROR_CODE_CATEGORY_NOT_VALID,
                ProcessingError::ERROR_LEVEL_NOT_CRITICAL,
                $rowData['rowNum'],
                self::COL_CATEGORY,
                __('Category "%1" has not been created.',
$error['category'])
                . ' ' . $error['exception']->getMessage()
            );
        }

        return $ids;
    }

    /**
     * @param $rowData
     * @return mixed
     * @throws NoSuchEntityException
     */
    protected function applyPriceRules($rowData)
    {
        if (!empty($this->_parameters['price_rules'])) {
            $priceRules = $this->_parameters['price_rules'];

            foreach ($priceRules as $priceRule) {
                $applyRule = true;

                if
(isset($priceRule['price_rules_conditions_hidden']['rule']['conditions
']])) {
                    $conditions =
$priceRule['price_rules_conditions_hidden']['rule']['conditions'];
                    $aggr = $conditions[1]['aggregator'];
                    $aggrValue = $conditions[1]['value'];
                    $data = [
                        'conditions' =>
$priceRule['price_rules_conditions_hidden']['rule']['conditions'],
                        'row' => $rowData,
                        'aggregator' => $aggr,
                        'value' => $aggrValue,
                        'categories' => isset($this-
>categoriesCache[$this->getCorrectSkuAsPerLength($rowData)]) ?
                            array_keys($this->categoriesCache[$this-
>getCorrectSkuAsPerLength($rowData)]) : [],
                    ];
                    if (empty($data['categories'])) {
                        $checkSku = $this-
>getCorrectSkuAsPerLength($rowData);
                        if (isset($this-
>_oldSku[strtolower($checkSku)])) {
                            /** @var \Magento\Catalog\Model\Product
$product */
                            $product = $this->productRepository-
>get($checkSku);
                            $data['categories'] = $product-
>getCategoryIds();
                        }
                    }
                    $applyRule = $this->priceRuleConditionFactory-
>create()->validatePriceRuleConditions($data);
                }

                if ($applyRule) {
                    if ($priceRule['apply'] == 'fixed') {
                        $rowData['price'] += $priceRule['value'];
                    } else {
                        $rowData['price'] *= 1 + $priceRule['value']
/ 100;
                    }
                }
            }
        }

        if (isset($this->_parameters['round_up_prices'],
$rowData['price'])
            && $this->_parameters['round_up_prices'] > 0) {
            $rowData['price'] = $this->roundPrice($rowData['price']);
        }

        if (isset($this->_parameters['round_up_special_price'],
$rowData['special_price'])
            && $this->_parameters['round_up_special_price'] > 0) {
            $rowData['special_price'] = $this-
>roundPrice($rowData['special_price']);
        }

        return $rowData;
    }

    /**
     * Round price to *.49 or to *.99
     *
     * @param $num
     *
     * @return float
     */
    protected function roundPrice($num): float
    {
        $fln = $num - floor($num);
        if ($fln > 0 && $fln < 0.5) {
            $fln = 0.49;
        } else {
            $fln = 0.99;
        }

        return floor($num) + $fln;
    }

    /**
     * @param array $data
     * @param string $rowSku
     *
     * @return array
     */
    protected function getTierPrices($data, $rowSku)
    {
        $tierPrices = [];

        if (!empty($data['tier_prices'])) {
            $tiers = explode("|", $data['tier_prices']);
            $groups = $this->groupFactory->create()->getCollection()-
>toOptionArray();
            $newGroups = [];
            foreach ($groups as $group) {
                $newGroups[$group['label']] = $group['value'];
            }
            $websites = $this->websiteFactory->create()-
>getCollection()->toOptionArray();
            $newWebsites = [0 => self::VALUE_ALL];
            foreach ($websites as $website) {
                $newWebsites[$website['label']] = $website['value'];
            }
            foreach ($tiers as $field) {
                $elements = array_map('trim', explode($this-
>getMultipleValueSeparator(), $field));
                $isAllGroup = 0;
                if ($elements[0] == __('ALL GROUPS')) {
                    $isAllGroup = 1;
                }
```

```
                if (version_compare($this->productMetadata-
>getVersion(), '2.2.0', '>=')) {
                    $tierPrices[$rowSku][] = [
                        'all_groups' => $isAllGroup,
                        'customer_group_id' => (isset($elements[0])
&& isset($newGroups[$elements[0]])) ?
                            $newGroups[$elements[0]] : 0,
                        'qty' => (isset($elements[1])) ? $elements[1]
: 0,
                        'value' => (isset($elements[2])) ?
$elements[2] : 0,
                        'percentage_value' => (isset($elements[3])) ?
                            (!empty($elements[3]) ? $elements[3] :
null) : null,
                        'website_id' => (isset($elements[4]) &&
isset($newWebsites[$elements[4]])) ?
                            $newWebsites[$elements[4]] : 0,
                    ];
                } else {
                    $tierPrices[$rowSku][] = [
                        'all_groups' => $isAllGroup,
                        'customer_group_id' => (isset($elements[0])
&& isset($newGroups[$elements[0]])) ?
                            $newGroups[$elements[0]] : 0,
                        'qty' => (isset($elements[1])) ? $elements[1]
: 0,
                        'value' => (isset($elements[2])) ?
$elements[2] : 0,
                        'website_id' => (isset($elements[3]) &&
isset($newWebsites[$elements[3]])) ?
                            $newWebsites[$elements[3]] : 0,
                    ];
                }
            }
        }
        return $tierPrices;
    }

    /**
     * @param $data
     * @param array $existingImages
     * @return $this
     */
    protected function saveConfigurationVariations($data,
$existingImages = [])
    {
        if (!empty($data)) {
            $simpleValAttr = [];
            if (isset($this->_parameters['copy_simple_value'])) {
                foreach ($this->_parameters['copy_simple_value'] as
$simpleValConfig) {
                    $simpleValAttr[] =
$simpleValConfig['copy_simple_value_attributes'];
                }
            }
            /**
             * @var string $skuConf
             * @var array $elements
             */
            foreach ($data as $skuConf => $elements) {
                $skuConf = (string) $skuConf;
                if (count($elements) < 1) {
                    continue;
                }
                if (version_compare($this->productMetadata-
>getVersion(), '2.2.0', '>=')) {
                    $checkSku = mb_strtolower($skuConf);
                }
                $categories = $websites = [];
                $additionalRows = [];
                $changeAttributes = [];
                $mediaGallery = [];
                $storeIds = $this->getStoreIds();
                $product = null;
                try {
                    $collection = $this->collectionFactory->create()
                        ->addFieldToFilter('sku', $skuConf)
                        ->addFieldToFilter('type_id', 'configurable')
                        ->addAttributeToSelect('*');
                    $this->addLogWriteln(__('Configure variations for
SKU:%1', $skuConf), $this->output, 'info');
                    if ($this->_parameters['configurable_create'] &&
!$collection->getSize()) {
                        try {
                            /** @var Collection $collectionChild */
                            $collectionChild = $this-
>collectionFactory->create();
                            $collectionChild->addFieldToFilter('sku',
$elements[0][self::COL_SKU])
                                ->addAttributeToSelect('*');
                            /** @var \Magento\Catalog\Model\Product
$child */
                            $child = $collectionChild-
>getFirstItem();
                            $data = [];
                            $data[self::COL_SKU] = $skuConf;
                            $data[self::COL_NAME] = $skuConf;
                            foreach ($simpleValAttr as $attr) {
                                $data[$attr] = $child-
>getData($attr);
                            }
                            $data['attribute_set_id'] = $child-
>getAttributeSetId();
                            $data['type_id'] = 'configurable';
                            $data['website_ids'] = $child-
>getWebsiteIds();
                            $websites[$skuConf] = $child-
>getWebsiteIds();
                            $data['category_ids'] = $child-
>getCategoryIds();
                            $data['visibility'] =
Visibility::VISIBILITY_BOTH;
                            $data['has_options'] = $child-
>getData('has_options');
                            $product = $this->_proxyProdFactory-
>create();
                            if ($this->_parameters['generate_url'] &&
isset($data[self::COL_NAME])) {
                                $storeIds = $this->getStoreIds();
                                $data = $this->generateUrlKey($data,
$storeIds);
                            }
                            $product->setData($data);
                            $product-
>setQuantityAndStockStatus(['qty' => 0, 'is_in_stock' => 1]);
                            $product = $this->productRepository-
>save($product);
                            $entityLinkField = $this-
>getProductEntityLinkField();
                            $data[$entityLinkField] = $product-
>getId();
                            $this->skuProcessor->addNewSku($skuConf,
$data);
                            $this->_oldSku[strtolower($skuConf)] = [
                                'type_id' => "configurable",
                                'attr_set_id' => $child-
>getAttributeSetId(),
                                $entityLinkField => $product-
>getId(),
                                'supported_type' => true,
                            ];
                            $productId = $product->getId();
                            foreach ($simpleValAttr as $attr) {
                                switch ($attr) {
                                    case
SystemOptions::RELATED_PRODUCT_ATTRIBUTE:
                                        $this-
>productLinkData[$productId]['type'][] =
ProductLink::LINK_TYPE_RELATED;
                                        break;
                                    case
SystemOptions::UP_SELLS_PRODUCT_ATTRIBUTE:
                                        $this-
>productLinkData[$productId]['type'][] =
ProductLink::LINK_TYPE_UPSELL;
                                        break;
                                    case
SystemOptions::CROSS_SELLS_PRODUCT_ATTRIBUTE:
                                        $this-
>productLinkData[$productId]['type'][] =
ProductLink::LINK_TYPE_CROSSSELL;
                                        break;
                                }
                            }
                        } catch (LocalizedException $e) {
                            $this->addLogWriteln($e->getMessage(),
$this->output, 'error');
                        }
                    } else {
                        if ($collection->getSize()) {
                            /** @var \Magento\Catalog\Model\Product
$product */
                            $product = $collection->getFirstItem();

                            if ($product->getSku() == $skuConf) {
                                /** @var Collection $collectionChild
*/
                                $collectionChild = $this-
>collectionFactory->create();
                                $collectionChild-
>addFieldToFilter('sku', $elements[0][self::COL_SKU])
                                    ->addAttributeToSelect('*');
                                /** @var
\Magento\Catalog\Model\Product $child */
                                $child = $collectionChild-
>getFirstItem();
                                $_updateData = [];
                                $productId = $product->getId();
                                foreach ($simpleValAttr as $attr) {
                                    switch ($attr) {
                                        case
SystemOptions::RELATED_PRODUCT_ATTRIBUTE:
                                            $this-
>productLinkData[$productId]['type'][] =
ProductLink::LINK_TYPE_RELATED;
                                            break;
                                        case
SystemOptions::UP_SELLS_PRODUCT_ATTRIBUTE:
                                            $this-
>productLinkData[$productId]['type'][] =
ProductLink::LINK_TYPE_UPSELL;
                                            break;
                                        case
SystemOptions::CROSS_SELLS_PRODUCT_ATTRIBUTE:
                                            $this-
>productLinkData[$productId]['type'][] =
```

```
ProductLink::LINK_TYPE_CROSSSELL;
                                        break;
                                    default:
                                        $_updateData[$attr] =
$child->getData($attr);
                                        break;
                                }
                            }
                            $objectManager =
ObjectManager::getInstance();
                            $productAction = $objectManager-
>create(Action::class);
                            $productAction-
>updateAttributes([$product->getId()], $_updateData, 0);
                            $websites[$skuConf] = $child-
>getWebsiteIds();
                            $childCategory = $child-
>getCategoryIds();
                            $configCat = $product-
>getCategoryIds();
                            $configCategory = [];
                            $del = true;
                            $deleteCat = array_diff($configCat,
$childCategory);
                            if (!isset($this-
>_parameters['remove_product_categories'])
                                && $this-
>_parameters['remove_product_categories'] > 0
                            ) {
                                $del = false;
                            }
                            if (!array_key_exists($skuConf,
$this->categoriesCache)) {
                                $this->categoriesCache[$skuConf]
= [];
                                $categories[$skuConf] = [];
                                $configCategory = [];
                            }
                            foreach (array_merge($childCategory,
$configCat) as $categoryId) {
                                if ($del === true &&
in_array($categoryId, $deleteCat, false)) {
                                    $configCategory[$categoryId]
= false;
                                } else {
                                    $configCategory[$categoryId]
= true;
                                }
                            }
                            $categories[$skuConf] =
$configCategory;
                        }
                        if ($product->getTypeId() !=
'configurable') {
                            $product->setTypeId('configurable');
                            $product = $this->productRepository-
>save($product);
                        }
                    } else {
                        $this->addLogWriteln(
                            __(
                                'Configurable Product for sku
"%1" not created before. Turn on feature to create ' .
                                'configurable product on the
fly',
                                $skuConf
                            ),
                            $this->getOutput(),
                            'error'
                        );

                        continue;
                    }
                }

                if (isset($this->_parameters['remove_images'])
                    && array_key_exists($skuConf,
$existingImages)
                    && $this->_parameters['remove_images'] == 1
                ) {
                    $this-
>removeExistingImages($existingImages[$skuConf]);
                    unset($existingImages[$skuConf]);
                }

                foreach ($this->_imagesArrayKeys as $fieldImage)
{
                    if ($fieldImage === '_media_image' ||
$fieldImage === ProductVideo::VIDEO_URL_COLUMN) {
                        continue;
                    }
                    $data[$fieldImage] = $child-
>getData($fieldImage);
                    $attributeChange = $this-
>retrieveAttributeByCode($fieldImage);
                    $attrId = $attributeChange->getId();
                    $attrTable = $attributeChange->getBackend()-
>getTable();
                    $attrValue = $child->getData($fieldImage);
                    if
(!isset($changeAttributes[$attrTable][$checkSku][$attrId][0]) &&
!empty($attrValue)) {

$changeAttributes[$attrTable][$skuConf][$attrId][0] = $attrValue;
                        if (version_compare($this-
>productMetadata->getVersion(), '2.2.4', '>=') ||
                            strpos($this->getProductMetadata()-
>getVersion(), '1.0.0') !== false) {

$mediaGallery[Store::DEFAULT_STORE_ID][$skuConf][] = [
                                'attribute_id' => $this-
>getMediaGalleryAttributeId(),
                                'label' => '',
                                'position' => 1,
                                'disabled' => '0',
                                'value' => $attrValue,
                            ];
                        } else {
                            $mediaGallery[$skuConf][] = [
                                'attribute_id' => $this-
>getMediaGalleryAttributeId(),
                                'label' => '',
                                'position' => 1,
                                'disabled' => '0',
                                'value' => $attrValue,
                            ];
                        }
                    }
                }

                $vars = [];
                $attributes = [];
                $attributeChange = $this-
>retrieveAttributeByCode('visibility');
                $attrTable = $attributeChange->getBackend()-
>getTable();

                $attrValue = 1;

                $attrId = $attributeChange->getId();
                foreach ($elements as $element) {
                    $position = 0;
                    foreach ($element as $ki => $field) {
                        if ($ki != 'sku' && !empty($field)) {
                            if (!in_array($ki, $attributes)) {
                                $attributes[] = $ki;
                            }
                            $vars['fields'][] = [
                                'code' => $ki,
                                'value' => $field,
                            ];
                        } else {
                            $vars[$ki] = $field;
                            if ($ki == 'sku') {
                                foreach ($storeIds as $storeId) {
                                    if
(!isset($changeAttributes[$attrTable][$field][$attrId][$storeId])) {

$changeAttributes[$attrTable][$field][$attrId][$storeId] = $attrValue;
                                    }
                                }
                            }
                        }
                    }
                    $vars['position'] = $position;
                    $position++;
                    $additionalRows[] = $vars;
                }
                $attributeValues = [];
                $ids = [];
                $configurableAttributesData = [];
                $position = 0;
                foreach ($attributes as $attribute) {
                    foreach ($additionalRows as $list) {
                        $attributeCollection = $this-
>attributeFactory->create()->getCollection();
                        $attributeCollection-
>addFieldToFilter('attribute_code', $attribute);
                        $value = [];
                        if (isset($list['fields'])) {
                            foreach ($list['fields'] as $item) {
                                if ($item['code'] == $attribute)
{
                                    $value = $item['value'];
                                    $collection = $this-
>collectionFactory->create();
                                    $collection-
>addFieldToFilter('sku', $list['sku']);
                                    if (!in_array($collection-
>getFirstItem()->getId(), $ids)) {
                                        $ids[] = $collection-
>getFirstItem()->getId();
                                    }
                                }
                            }
                        }
                        if ($attributeCollection->getSize()) {
                            $attributeValues[$attribute][] = [
                                'label' => $attribute,
                                'attribute_id' =>
$attributeCollection->getFirstItem()->getId(),
                                'value_index' => $value,
                            ];
                        }
                    }
                    if ($attributeCollection->getSize()) {
                        $attr = $attributeCollection-
>getFirstItem();
                        $configurableAttributesData[] =
```

```
                                [
                                    'attribute_id' => $attr->getId(),
                                    'code' => $attr-
>getAttributeCode(),
                                    'label' => $attr-
>getStoreLabel(),
                                    'position' => $position++,
                                    'values' =>
$attributeValues[$attribute],
                                ];
                        }
                    }

                    /**
                     * Check if attributes was added to target
attribute set.
                     */
                    if (isset($product) && $product-
>getAttributeSetId() > 0) {
                        $invalidAttributes = [];
                        $attributeSetId = $product-
>getAttributeSetId();
                        foreach ($configurableAttributesData as
$attribute) {
                            $attributeId =
$attribute['attribute_id'];
                            $select = $this->_connection->select()
                                ->from(
                                    $this->getResource()-
>getTable('eav_entity_attribute'),
                                    'attribute_id'
                                )->where(
                                    'attribute_set_id = ?',
                                    $attributeSetId
                                )->where(
                                    'attribute_id = ?',
                                    $attributeId
                                );
                            $result = $this->_connection-
>fetchCol($select);
                            if (empty($result)) {
                                $invalidAttributes[] =
$attribute['code'];
                            }
                        }
                        if (!empty($invalidAttributes)) {
                            throw new LocalizedException(
                                __(
                                    "Attributes '%1' is not attached
to related attribute set.",
                                    implode(', ', $invalidAttributes)
                                )
                            );
                        }
                    }
                    if (!empty($mediaGallery)) {
                        $this->_saveMediaGallery($mediaGallery);
                    }
                    if (!empty($websites)) {
                        $this->_saveProductWebsites($websites,
$product->getId(), true);
                    }
                    if (!empty($categories)) {
                        $this->_saveProductCategories($categories,
$product->getId(), true);
                    }
                    if (!empty($changeAttributes)) {
                        $this-
>_saveProductAttributes($changeAttributes);
                    }
                    if ($product !== null) {
                        $this->saveCollectData($product,
$configurableAttributesData, $ids);
                    }
                } catch (Exception $e) {
                    $this->getErrorAggregator()->addError(
                        $e->getCode(),
                        ProcessingError::ERROR_LEVEL_NOT_CRITICAL,
                        null,
                        null,
                        $e->getMessage()
                    );
                }
            }
        }

        return $this;
    }

    /**
     * Add related, cross-sell or up-sell products
     *
     * @param $configurableProductId
     * @param $typesLink
     * @throws LocalizedException
     */
    protected function addProductLinks($configurableProductId,
$typesLink)
    {
        $resource = $this->_linkFactory->create();
        $mainTable = $this->productLink->getLinkCollection()-
>getMainTable();
        $nextLinkId = $this->_resourceHelper-
>getNextAutoincrement($mainTable);
        $allLinkedProducts = [];
        $positionRows = [];
        $select = $this->_connection->select()
            ->from($this->getResource()-
>getTable('catalog_product_relation'))
            ->where('parent_id=?', $configurableProductId);
        $simpleProductId = $this->_connection-
>fetchRow($select)['child_id'];
        $select = $this->_connection->select()
            ->from($mainTable)
            ->reset(Zend_Db_Select::COLUMNS)
            ->columns(['linked_product_id', 'link_type_id'])
            ->where('product_id=?', $simpleProductId);
        $linkedProducts = $this->_connection->fetchAll($select);
        $this->_connection->delete(
            $mainTable,
            "product_id=$configurableProductId"
        );
        $i = 0;
        foreach ($linkedProducts as $product) {
            if (in_array($product['link_type_id'],
$typesLink['type'])) {
                $product['product_id'] = $configurableProductId;
                $product['link_id'] = $nextLinkId;
                $allLinkedProducts[] = $product;
                $positionRows[] = [
                    'link_id' => $nextLinkId,
                    'product_link_attribute_id' =>
$product['link_type_id'],
                    'value' => $i++,
                ];
                $nextLinkId++;
            }
        }
        if (isset($allLinkedProducts) && !empty($allLinkedProducts))
{
            $this->_connection->insertMultiple(
                $mainTable,
                $allLinkedProducts
            );
            $this->_connection->insertOnDuplicate(
                $resource->getAttributeTypeTable('int'),
                $positionRows,
                ['value']
            );
        }
    }

    /**
     * @param array $websiteData
     * @param null $productId
     * @param bool $config
     *
     * @return $this|MagentoProduct
     */
    protected function _saveProductWebsites(array $websiteData,
$productId = null, $config = false)
    {
        static $productWebsiteTable = null;
        $removeWebsite = $this->_parameters['remove_product_website']
?? 0;
        if ($removeWebsite || $productId) {
            if (!$productWebsiteTable) {
                $productWebsiteTable = $this->getResource()-
>getProductWebsiteTable();
            }
            if ($websiteData) {
                $newWebsiteData = [];
                $deletedProductIds = [];

                foreach ($websiteData as $productSku =>
$productWebsites) {
                    if ($this->onlyUpdate
                        && $removeWebsite > 0
                        && empty($productWebsites)
                    ) {
                        $this->addLogWriteln(
                            __('Product %1 Website Cannot be
Cleared', $productSku),
                            $this->output,
                            'info'
                        );
                        continue;
                    }
                    if (!$config) {
                        $productId = $this->skuProcessor-
>getNewSku($productSku)['entity_id'];
                    }
                    $deletedProductIds[] = $productId;
                    if ($config) {
                        foreach ($productWebsites as $websiteId) {
                            $newWebsiteData[] = ['product_id' =>
$productId, 'website_id' => $websiteId];
                        }
                    } else {
                        foreach (array_keys($productWebsites) as
$websiteId) {
                            $newWebsiteData[] = ['product_id' =>
$productId, 'website_id' => $websiteId];
                        }
                    }
                }
                $this->_connection->delete(
                    $productWebsiteTable,
                    $this->_connection->quoteInto('product_id IN
(?)', $deletedProductIds)
```

```
                );

                if ($newWebsiteData) {
                    $this->_connection->insertOnDuplicate($productWebsiteTable, $newWebsiteData);
                }
            }
            return $this;
        }
        return parent::_saveProductWebsites($websiteData);
    }

    /**
     * @param \Magento\Catalog\Model\Product $product
     * @param array $configurableAttributesData
     * @param array $ids
     * @throws Exception
     */
    public function saveCollectData($product, $configurableAttributesData, $ids)
    {
        $productId = $product->getData($this->getProductEntityLinkField());

        $connection = $this->_connection;
        $resource = $this->getResource();
        $table = $resource->getTable('catalog_product_super_attribute');
        $labelTable = $resource->getTable('catalog_product_super_attribute_label');
        $linkTable = $resource->getTable('catalog_product_super_link');
        $relationTable = $resource->getTable('catalog_product_relation');
        $select = $connection->select()->from(
            ['m' => $table],
            ['product_id', 'attribute_id', 'product_super_attribute_id']
        )->where(
            'm.product_id IN ( ? )',
            [$productId]
        );

        //check if import source has different product variation for this product then delete current variations
        $counts = count($connection->fetchAll($select));
        if ($counts) {
            $isChangeOfAttributeVariation = false;
            if ($counts != count($configurableAttributesData)) {
                $isChangeOfAttributeVariation = true;
            } else {
                $currentattrConfig = $connection->fetchAll($select);
                $currentAttributeIds = [];
                if (!empty($currentattrConfig)) {
                    foreach ($currentattrConfig as $key => $value) {
                        $currentAttributeIds[] = $value['attribute_id'];
                    }
                }

                $newAttrIds = [];
                foreach ($configurableAttributesData as $elem) {
                    $newAttrIds[] = $elem['attribute_id'];
                }
                sort($currentAttributeIds);
                sort($newAttrIds);
                if ($currentAttributeIds != $newAttrIds) {
                    $isChangeOfAttributeVariation = true;
                }
            }
            if ($isChangeOfAttributeVariation) {
                $whereConditions = [
                    $connection->quoteInto('product_id = ?', $productId),
                ];
                $deleteRows = $connection->delete($table, $whereConditions);
            }
        }

        //inser new product variation for configurable product
        $counts = count($connection->fetchAll($select));
        if (!$counts) {
            foreach ($configurableAttributesData as $elem) {
                $data = [
                    'product_id' => $productId,
                    'attribute_id' => $elem['attribute_id'],
                    'position' => $elem['position'],
                ];
                $connection->insertOnDuplicate($table, $data);
            }

            foreach ($connection->fetchAll($select) as $row) {
                $attrId = $row['attribute_id'];
                $superId = $row['product_super_attribute_id'];
                foreach ($configurableAttributesData as $elem) {
                    if ($elem['attribute_id'] == $attrId) {
                        $data = ['product_super_attribute_id' => $superId, 'value' => $elem['label']];
                        $connection->insertOnDuplicate($labelTable, $data);
                    }
                }
            }
        }

        $first = 0;
        foreach ($ids as $id) {
            $data = ['product_id' => $id, 'parent_id' => $productId];
            $connection->insertOnDuplicate($linkTable, $data);
            if ($this->manager->isEnabled('Ivan_ConfigurableProducts') && !$first) {
                $connection->insertOnDuplicate(
                    $resource->getTable('icp_catalog_product_default_super_link'),
                    $data
                );
                $first = 1;
            }
            $relData = ['child_id' => $id, 'parent_id' => $productId];
            $connection->insertOnDuplicate($relationTable, $relData);
        }
    }

    /**
     * @param string $sku
     * @return bool
     */
    public function isExist($sku)
    {
        if ($this->onlyUpdate) {
            $collectionUpdate = $this->collectionFactory->create()->addFieldToFilter(
                self::COL_SKU,
                $sku
            );
            if (!$collectionUpdate->getSize()) {
                return true;
            }
        }

        return false;
    }

    /**
     * Load categories map
     *
     * @param string $fieldName
     * @return array
     * @throws LocalizedException
     */
    public function getCategoriesMap($fieldName)
    {
        $bunchRows = [];
        $categories = [];
        $source = $this->_getSource();
        $source->rewind();
        $i = 1;
        while ($source->valid() || $bunchRows) {
            if ($source->valid()) {
                $rowData = $source->current();
                if (isset($rowData[$fieldName])) {
                    $categories[] = $rowData[$fieldName];
                }
                $i++;
                $source->next();
            }
        }

        return $categories;
    }

    /**
     * Validate data
     *
     * @param int $saveBunches
     * @return ProcessingErrorAggregatorInterface
     * @throws LocalizedException
     * @throws Zend_Validate_Exception
     * @throws Exception
     */
    public function validateData($saveBunches = 1)
    {
        if ($this->_parameters['behavior'] == Import::Ivan_ONLY_UPDATE) {
            $this->onlyUpdate = 1;
            $this->_parameters['behavior'] = Import::BEHAVIOR_APPEND;
        } elseif ($this->_parameters['behavior'] == Import::Ivan_ONLY_ADD) {
            $this->onlyAdd = true;
            $this->_parameters['behavior'] = Import::BEHAVIOR_APPEND;
        }

        if (isset($this->_parameters['output'])) {
            $this->output = $this->_parameters['output'];
        }

        $this->_initTypeModels();
        if (!$this->_dataValidated) {
            $this->getErrorAggregator()->clear();
            // do all permanent columns exist?
            $platformModel = null;
            $absentColumns = array_diff($this->_permanentAttributes, $this->getSource()->getColNames());
            $this->addErrors(self::ERROR_CODE_COLUMN_NOT_FOUND, $absentColumns);

            // check attribute columns names validity
            $columnNumber = 0;
```

```
            $emptyHeaderColumns = [];
            $invalidColumns = [];
            $invalidAttributes = [];
            foreach ($this->getSource()->getColNames() as
$columnName) {
                $this->addLogWriteln(__('Checked column %1',
$columnNumber), $this->output);
                $columnNumber++;
                if (!$this->isAttributeParticular($columnName)) {
                    /**
                     * Check syntax when attribute should be created
on the fly
                     */
                    $createValuesAllowed = (bool)$this->scopeConfig-
>getValue(
                        Import::CREATE_ATTRIBUTES_CONF_PATH,
                        ScopeInterface::SCOPE_STORE
                    );
                    if ($createValuesAllowed &&
preg_match('/^(attribute\|).+/', $columnName)) {
                        $attrCodes = [];
                        $columnData = explode('|', $columnName);
                        $columnData = $this-
>prepareAttributeData($columnData);
                        $attribute = $this->attributeFactory-
>create();
                        $attribute-
>loadByCode(\Magento\Catalog\Model\Product::ENTITY,
$columnData['attribute_code']);
                        if (!$attribute->getId()) {
                            $this-
>prepareAttributesData($columnData);
                            $attribute->setBackendType(
                                $attribute-
>getBackendTypeByInput($columnData['frontend_input'])
                            );
                            $defaultValueField = $attribute-
>getDefaultValueByInput($columnData['frontend_input']);
                            if (!$defaultValueField &&
isset($columnData['default_value'])) {
                                unset($columnData['default_value']);
                            }
                            $columnData['source_model'] = $this-
>productHelper->getAttributeSourceModelByInputType(
                                $columnData['frontend_input']
                            );
                            $columnData['backend_model'] = $this-
>productHelper->getAttributeBackendModelByInputType(
                                $columnData['frontend_input']
                            );

                            $attribute->addData($columnData);
                            try {
                                $attribute->save();
                            } catch (Exception $e) {
                                $invalidColumns[] = $columnName;
                            }
                            $attrSetCodes = explode(',',
$columnData[self::ATTRIBUTE_SET_COLUMN]);
                            foreach ($attrSetCodes as $attrSetCode) {
                                if (isset($this-
>_attrSetNameToId[$attrSetCode])) {
                                    $attrSetId = $this-
>_attrSetNameToId[$attrSetCode];
                                    $attrGroupCode =
isset($columnData[self::ATTRIBUTE_SET_GROUP])
                                        ?
$columnData[self::ATTRIBUTE_SET_GROUP] : 'product-details';
                                    if (!isset($this-
>_attributeSetGroupCache[$attrSetId])) {
                                        $groupCollection =
                                            $this-
>groupCollectionFactory->create()->setAttributeSetFilter(
                                                $attrSetId
                                            )->load();
                                        foreach ($groupCollection as
$group) {
                                            $attributeGroupCode =
$group->getAttributeGroupCode();
                                            $this-
>_attributeSetGroupCache[$attrSetId][$attributeGroupCode] =
                                                $group-
>getAttributeGroupId();
                                        }
                                    }
                                    foreach ($this-
>_attributeSetGroupCache[$attrSetId] as $groupCode => $groupId) {
                                        if ($groupCode ==
$attrGroupCode) {
                                            $attribute-
>setAttributeSetId($attrSetId);
                                            $attribute-
>setAttributeGroupId($groupId);
                                            try {
                                                $attribute->save();
                                                $attrCodes[] =
$attribute->getAttributeCode();
                                            } catch (Exception $e) {
                                                $this-
>addLogWriteln($e->getMessage(), $this->output, 'error');
                                            }
                                            break;
                                        }
                                    }
                                }
                            }
                        }
                        $this->_specialAttributes =
array_merge($this->_specialAttributes, $attrCodes);
                    }

                    if (trim($columnName) == '') {
                        $emptyHeaderColumns[] = $columnNumber;
                    } elseif ($this->needColumnCheck &&
!in_array($columnName, $this->getValidColumnNames())) {
                        $invalidAttributes[] = $columnName;
                    }
                }
            }

            $this->addErrors(self::ERROR_CODE_INVALID_ATTRIBUTE,
$invalidAttributes);
            $this->addErrors(self::ERROR_CODE_COLUMN_EMPTY_HEADER,
$emptyHeaderColumns);
            $this->addErrors(self::ERROR_CODE_COLUMN_NAME_INVALID,
$invalidColumns);
            $this->addLogWriteln(__('Finish checking columns'),
$this->output);
            $this->addLogWriteln(__('Errors count: %1', $this-
>getErrorAggregator()->getErrorsCount()), $this->output);
            if (!$this->getErrorAggregator()->getErrorsCount()) {
                if ($saveBunches) {
                    $this->addLogWriteln(__('Start saving bunches'),
$this->output);
                    $this->mergeFieldsMap();

                    $platform = null;
                    $platformName = $this->_parameters['platforms']
?? null;
                    if ($platformName && is_string($platformName) &&
$platformName != 'magento2') {
                        $platform = $this->helper-
>getPlatformModel($platformName);
                    }
                    /* use platform @todo remove this */
                    if ($platform && method_exists($platform,
'saveValidatedBunches')) {
                        $platform->saveValidatedBunches(
                            $this->_getSource(),
                            $this->_resourceHelper->getMaxDataSize(),
                            $this->_importExportData->getBunchSize(),
                            $this->_dataSourceModel,
                            $this->_parameters,
                            $this->getEntityTypeCode(),
                            $this->getBehavior(),
                            $this->_processedRowsCount,
                            $this->getMultipleValueSeparator(),
                            $this
                        );
                    } else {
                        $this->_saveValidatedBunches();
                    }

                    $this->addLogWriteln(__('Finish saving bunches'),
$this->output);
                    $this->_dataValidated = true;
                }
            }
        }

        return $this->getErrorAggregator();
    }

    /**
     * @return $this
     * @throws LocalizedException
     */
    protected function _initTypeModels()
    {
        $this->_importConfig = $this->fireImportConfig;
        $productTypes = $this->_importConfig->getEntityTypes($this-
>getEntityTypeCode());
        foreach ($productTypes as $productTypeName =>
$productTypeConfig) {
            $class = $productTypeConfig['model'];
            $class::$commonAttributesCache = [];
            $class::$attributeCodeToId = [];
        }

        parent::_initTypeModels();

        return $this;
    }

    /**
     * Checks and sets appropriate data for swatch attribute
     *
     * @param array $data
     */
    protected function prepareAttributesData(&$data)
    {
        if (isset($data['frontend_input'])) {
            switch ($data['frontend_input']) {
                case 'swatch_visual':
                    $data[Swatch::SWATCH_INPUT_TYPE_KEY] =
Swatch::SWATCH_INPUT_TYPE_VISUAL;
                    $data['frontend_input'] = 'select';
                    break;
                case 'swatch_text':
```

```
                    $data[Swatch::SWATCH_INPUT_TYPE_KEY] =
Swatch::SWATCH_INPUT_TYPE_TEXT;
                    $data['use_product_image_for_swatch'] = 0;
                    $data['frontend_input'] = 'select';
                    break;
                case 'select':
                    $data[Swatch::SWATCH_INPUT_TYPE_KEY] =
Swatch::SWATCH_INPUT_TYPE_DROPDOWN;
                    $data['frontend_input'] = 'select';
                    break;
            }
        }
    }

    /**
     * mergeFieldsMap function
     */
    protected function mergeFieldsMap()
    {
        if (isset($this->_parameters['map'])) {
            $newAttributes = [];
            foreach ($this->_parameters['map'] as $field) {
                if (!$field['import'] || !empty($field['import'])) {
                    $field['import'] = $field['system'];
                }
                $attribute = $this->getResource()-
>getAttribute($field['system']);
                if ($attribute) {
                    $newAttributes[$attribute->getAttributeCode()] =
$field['import'];
                } else {
                    $newAttributes[$field['system']] =
$field['import'];
                }
            }

            $this->_fieldsMap = array_merge($this->_fieldsMap,
$newAttributes);
        }
    }

    /**
     * @return $this|MagentoProduct
     * @throws LocalizedException
     * @throws Zend_Validate_Exception
     * @throws Exception
     */
    protected function _saveValidatedBunches()
    {
        $_currentRowSkus = [];
        $source = $this->_getSource();
        $currentDataSize = 0;
        $bunchRows = [];
        $startNewBunch = false;
        $nextRowBackup = [];
        $maxDataSize = $this->_resourceHelper->getMaxDataSize();
        $bunchSize = $this->_importExportData->getBunchSize();
        $skuSet = [];

        $source->rewind();
        $this->_dataSourceModel->cleanBunches();
        $file = null;
        $jobId = null;
        if (isset($this->_parameters['file'])) {
            $file = $this->_parameters['file'];
        }
        if (isset($this->_parameters['job_id'])) {
            $jobId = $this->_parameters['job_id'];
        }

        while ($source->valid() || $bunchRows) {
            if ($startNewBunch || !$source->valid()) {
                if (version_compare($this->productMetadata-
>getVersion(), '2.2.4', '>=')) {
                    $bunchRows = $this-
>prepareCustomOptionRows($bunchRows);
                }
                if (empty($this->_parameters['generate_url'])) {
                    $bunchRows = $this-
>findUrlKeyDuplicates($bunchRows);
                }
                $this->addLogWriteln(__('Saving Validated Bunches'),
$this->output, 'info');

                $this->_dataSourceModel->saveBunches(
                    $this->getEntityTypeCode(),
                    $this->getBehavior(),
                    $jobId,
                    $file,
                    $bunchRows
                );
                $bunchRows = $nextRowBackup;
                $currentDataSize = strlen($this->getSerializer()-
>serialize($bunchRows));
                $startNewBunch = false;
                $nextRowBackup = [];
            }
            if ($source->valid()) {
                try {
                    $rowData = $source->current();
                    $isCached = $this->_parameters['cache_products']
?? false;
                    if ($isCached) {
                        $currentRowHash = sha1(trim(implode('',
$rowData)));
                        $cache = $this->cache->load($currentRowHash);
                        if ($cache) {
                            $this->addLogWriteln(
                                __('Product %1 has not changed',
$rowData['sku']),
                                $this->output,
                                'info'
                            );
                            $source->next();
                            continue;
                        }
                        $this-
>originalImportRows[strtolower($rowData['sku'])][] = $currentRowHash;
                    }
                    if (array_key_exists('sku', $rowData)) {
                        $skuSet[$rowData['sku']] = true;
                    }
                    $invalidAttr = [];
                    foreach ($rowData as $attrName => $element) {
                        if (is_string($element)) {
                            if (!mb_check_encoding($element, 'UTF-
8')) {
                                unset($rowData[$attrName]);
                                $invalidAttr[] = $attrName;
                            }
                        }
                    }
                    if (!empty($invalidAttr)) {
                        $this->addRowError(

AbstractEntity::ERROR_CODE_ILLEGAL_CHARACTERS,
                            $this->_processedRowsCount,
                            implode(',', $invalidAttr)
                        );
                    }
                } catch (InvalidArgumentException $e) {
                    $this->addRowError($e->getMessage(), $this-
>_processedRowsCount);
                    $this->_processedRowsCount++;
                    $source->next();
                    continue;
                }
                $rowData = array_map('trim', $rowData);
                if (isset($rowData['configurable_variations']) &&
$rowData['configurable_variations']) {
                    $this-
>checkAttributePresenceInAttributeSet($rowData);
                }
                $rowData[self::COL_SKU] = $this-
>getCorrectSkuAsPerLength($rowData);
                $_currentRowSkus[] =
mb_strtolower($rowData[self::COL_SKU]);
                $rowData = $this->customFieldsMapping($rowData);
                $rowData = $this->customBunchesData($rowData);

                $this->_processedRowsCount++;

                if ($this->onlyUpdate || $this->onlyAdd) {
                    $oldSkus = $this->skuProcessor->getOldSkus();
                    $productSku = strtolower($this-
>getCorrectSkuAsPerLength($rowData));

                    if (!isset($oldSkus[$productSku]) && $this-
>onlyUpdate) {
                        $source->next();
                        continue;
                    } elseif (isset($oldSkus[$productSku]) && $this-
>onlyAdd) {
                        $source->next();
                        continue;
                    }
                }

                if ($this->onlyUpdate) {
                    foreach ($rowData as $key => $value) {
                        if ('' == $value) {
                            unset($rowData[$key]);
                        }
                    }
                }

                if ($this->getBehavior() == Import::BEHAVIOR_REPLACE)
{
                    if (isset($rowData['attribute_set_code'])) {
                        $rowData['_attribute_set'] =
$rowData['attribute_set_code'];
                    }
                }
                $rowData = $this->processUrlKey($rowData);
                /* specify url_key for to avoid product repository
load
                in
\Magento\CatalogUrlRewrite\Observer\AfterImportDataObserver*/
                if (empty($rowData[self::URL_KEY])) {
                    $rowData[self::URL_KEY] = $this->onlyUpdate ? ''
: $this->getUrlKey($rowData);
                }

                if ($this->validateRow($rowData, $source->key())) {
                    // add row to bunch for save
                    $rowData = $this->_prepareRowForDb($rowData);
                    $rowSize = strlen($this->getSerializer()-
>serialize($rowData));
```

```
                    $isBunchSizeExceeded = $bunchSize > 0 &&
count($bunchRows) >= $bunchSize;

                    if (!$this->validateRowByProductType($rowData,
$source->key())) {
                        $this-
>addRowError(ValidatorInterface::ERROR_TYPE_UNSUPPORTED, $source-
>key());
                    }

                    if (($rowData['sku'] !== $this->getLastSku()) &&
                        ($currentDataSize + $rowSize >= $maxDataSize
|| $isBunchSizeExceeded)) {
                        $startNewBunch = true;
                        $nextRowBackup = [$source->key() =>
$rowData];
                    } else {
                        $bunchRows[$source->key()] = $rowData;
                        $currentDataSize += $rowSize;
                    }
                    $this->setLastSku($rowData['sku']);
                }

                $source->next();
            }
        }
        if (isset($this->_parameters['disable_products']) && $this-
>_parameters['disable_products'] > 0) {
            $this->disableProductsNotInList($_currentRowSkus);
        }
        $this->getOptionEntity()->validateAmbiguousData();
        $this->_processedEntitiesCount = (count($skuSet)) ?: $this-
>_processedRowsCount;

        $this->cache->clean([ImportProductCache::BUFF_CACHE]);
        $this->cache->save(
            $this->getSerializer()->serialize($this-
>originalImportRows),
            sha1(ImportProductCache::BUFF_CACHE)
        );

        return $this;
    }

    /**
     * @param $rowData
     * @return array
     * @throws Exception
     */
    public function processUrlKey($rowData)
    {
        if ($this->isSkuExist($rowData[self::COL_SKU])) {
            $rowData[self::URL_KEY] = $this->urlKeyManager
                ->getUrlKeyForSku($rowData[self::COL_SKU]);
        }
        if ($this->_parameters['generate_url'] == 1
            && $this->isNeedToChangeUrlKey($rowData)
        ) {
            $rowData = $this->generateUrlKey($rowData);
        }
        return $rowData;
    }

    /**
     * @param array $_currentRowSkus
     * @throws Exception
     */
    public function disableProductsNotInList(array $_currentRowSkus)
    {
        $attributeCode = $this->scopeConfig-
```

Mon Jun 05 20:27:25 EEST 2023, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 3.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 10%**

ID: 114871 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА Додано в БД: 2023-06-05 Автора: І.О. Андросюк Керівники: О.А.Пасічник Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	59650	904	2475 (4%)	40 (4%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1015443024

Дата перевірки:
05.06.2023 21:20:47 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
05.06.2023 21:21:22 EEST

ID користувача:
100005671

Назва документа: **КНс-20-1 Андросюк**

Кількість сторінок: **60** Кількість слів: **9087** Кількість символів: **70897** Розмір файлу: **2.40 MB** ID файлу: **1015103568**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

4.72%
Схожість

Найбільша схожість: **1.61%** з Інтернет-джерелом (https://openarchive.nure.ua/bitstream/document/17733/1/2021_M_ST_..

4.46% Джерела з Інтернету	312	Сторінка 62
1.89% Джерела з Бібліотеки	112	Сторінка 63

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	8
Підозріле форматування	17 сторінок

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНИХ НАУК

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Технологія трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів

Автор: Андросюк Іван Олександрович

Спеціальність: 122 – Комп'ютерні науки

Освітня програма: освітньо-професійна

Науковий керівник: к.т.н., доцент Пасічник Олександр Анатолійович

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) за програмою Anti-Plagiarism виявлені 3 %, схожість виявлена зі звітом автора з науково-дослідної практики. Найбільша схожість: 1.61% з Інтернет-джерелом (https://openarchive.nure.ua/bitstream/document/17733/1/2021_M_ST_…, яке містить матеріали огляду предметної області; інші схожості є фрагментарними – містять поширені конструкції, загальновідомі терміни, скорочення та визначення.

збігів/ідентичності/схожості, складає 3 % і 4.72% відповідно, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи Олександр ПАСІЧНИК

Гарант ОП Олександр МАЗУРЕЦЬ

Завідувач кафедри КН Олександр БАРМАК

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КНс-20-1 Андросюка Івана Олександровича*

за темою *Технологія трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів*

1. Актуальність теми

Інтенсивний розвиток, зростання функціональності, розширення сфер застосування та поглиблення інтелектуалізації вирішуваних задач при впровадженні програмних систем та технологій є стійкою та головною тенденцією розвитку людства останніми десятиліттями та залишиться такою в досяжному майбутньому. Однією з сфер діяльності, яка є флагманом у використанні цифрових технологій виступає торгівля. Наявні величезні успіхи та досягнення в цій області висувають все нові виклики для забезпечення стійкого росту галузі. Побічним ефектом цих успіхів та цього зростання, який необхідно подолати, є ефективне керування все зростаючою кількістю замовлень із дотримання прийнятних часових параметрів їх опрацювання. Шляхами вирішення цієї задачі є використання сучасних платформ у поєднанні із заходами покращення структури потоків інформації в межах наявних рішень.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

За стандартом, а саме описом предметної області, об'єктами вивчення та діяльності є математичні та інформаційні моделі реальних систем і процесів та методи і технології отримання, зберігання, обробки, передачі та використання інформації. Метою роботи є розробка технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів розробка інформаційної системи реалізації вказаної технології. При вирішенні поставленої задачі використано математичні моделі, методи та алгоритми розв'язання теоретичних і прикладних задач, що виникають при розробці інформаційних технологій. Тому результати виконання кваліфікаційної роботи бакалавра відповідають стандарту бакалавра спеціальності 122 – Комп'ютерні науки.

3. Професійні та особистісні якості бакалавра

При роботі над кваліфікаційною роботою бакалавра Боровик Дмитро Олегович проявив себе кваліфікованим фахівцем та дисциплінованим студентом, вчасно виконуючи

поставлені етапи дослідження. При написанні пояснювальної записки та при розробці прикладного програмного забезпечення проявив достатні для одержання успішного результату загальні та фахові компетентності та досягнуті програмні результати навчання за напрямком «Комп'ютерні науки».

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Одержані в роботі результати є наслідком особистої діяльності студента, який самостійно виконував всі поставлені задачі.

5. Ступінь оволодіння методами дослідження

При реалізації кваліфікаційної роботи показав достатній рівень компетентностей та володіння необхідним інструментарієм, методами, методиками та технологіями предметної області комп'ютерних наук.

6. Повнота та якість розкриття теми роботи

Тема роботи в повній мірі обґрунтована й розкрита, проведено аналіз актуальності та відомих досліджень в межах обраної теми, поставлені завдання, які у роботі виконані, та розроблено програмне забезпечення для апробації запропонованої технології.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Структура роботи та послідовність викладення логічні та відповідають поставленій меті. Викладення матеріалу послідовне, аргументоване, літературно грамотне.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Розроблена у роботі технологія та її програмна реалізація може бути використана в сфері електронної комерції для широкої номенклатури товарів для підвищення ефективності ведення господарської діяльності підприємствами та установами всіх форм власності.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

Враховуючи високий рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка ***«відмінно»*** *.*

Керівник _____ *к.т.н., доцент Олександр ПАСІЧНИК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КНс-20-1 Андросюка Івана Олександровича*
за темою: *Технологія трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів*

1. Актуальність обраної теми

Однією з сфер діяльності, яка є флагманом у використанні цифрових технологій виступає торгівля. Величезні успіхи та досягнення в цій області обумовлені як загальноцивілізаційними трендами, так, й викликами сьогодення на кшталт пандемії короновірсу, все одно висувають все нові вимоги для забезпечення стійкого росту галузі. Як результат цих успіхів та цього зростання, на який слід обов'язково слід зважати, є ефективне керування все зростаючою кількістю замовлень із дотримання прийнятних часових норм їх обробки.

2. Повнота розкриття мети та завдань роботи

Мета роботи розкрита повністю, всі завдання виконані.

3. Зміст кожного розділу роботи

В першому розділі виконано аналіз сучасних платформ електронної комерції та сучасних тенденцій покращення їх роботи. Визначено мету роботи та виконано постановку завдань.
В другому розділі реалізовано Технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів із зазначення принципів та підходів їх поділу та подальшого агрегатування.
В третьому розділі виконано Програмна реалізація технології трансферу даних для платформи Magento 2 e-commerce з використанням декомпозиції запитів та проведено експериментальне тестування.

4. Оцінка розробленої інформаційної системи, її практична цінність

Розроблена у роботі технологія та її програмна реалізація може бути використана в сфері електронної комерції для будь якої номенклатури товарів в умовах використання платформи Magento 2 e-commerce .

5. Якість оформлення кваліфікаційної роботи бакалавра

Структура роботи та послідовність викладення логічні та відповідають поставленій меті. Викладення матеріалу послідовне, аргументоване, літературно грамотне.

6. Недоліки кваліфікаційної роботи бакалавра

З роботи не зрозуміло, чи може бути поширена реалізована технологія на інші платформи електронної комерції

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка ***«відмінно»***.

Рецензент Доцент ТНІТ Олег ПИВОВАР