МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 42.00.00.000 ПЗ

Група ІПм-23-1

Дюг Андрій

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Дюг Андрій Іванович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

МАГІСТЕРСЬКА РОБОТА

Моделі, методи та алгоритми побудови журналу подій для хмарних систем

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Дюг А.І.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Бандура Вікторія Валеріївна, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Нормоконтроль

доц. Вовк Р.Б.

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2024 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Дюгу Андрію Івановичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Моделі, методи та алгоритми побудови журналу подій для хмарних систем”

керівник проекту (роботи) Бандура Вікторія Валеріївна, к.т.н., доцент

затверджені наказом закладу вищої освіти від “ 22 ” листопада 2024 р. № 781/7

2. Строк подання студентом проекту (роботи) 15 грудня 2024 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та формальні моделі побудови та функціонування інформаційних технологій роботи хмарних систем

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Дослідження предметної області обробки даних журналів подій в хмарних системах
2. Методи, методика та вимоги до систем обробки та видобування даних в журналах подій
3. Імплементація методів видобування даних для побудови журналу подій хмарних систем
4. Реалізація отримання даних з журналу подій на основі артефакто-орієнтованого підходу

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Візуальне представлення етапів процесів видобутку даних в журналах подій (рис. 1.1)
2. Підхід аналізу даних журналу подій (рис. 1.2)
3. Візуальне представлення Process Mining (рис. 1.3)
4. Приклад журналу подій (рис. 1.4)
5. Типи журналів подій (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2024 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури по темі магістерської роботи	15.09.2024	виконано
2	Аналіз концепцій та алгоритмів предметної області	29.09.2024	виконано
3	Дослідження предметної області обробки даних журналів подій в хмарних системах	15.10.2024	виконано
4	Методи, методика та вимоги до систем обробки та видобування даних в журналах подій	08.11.2024	виконано
5	Імплементація методів видобування даних для побудови журналу подій хмарних систем	20.11.2024	виконано
6	Реалізація отримання даних з журналу подій на основі артефакто-орієнтованого підходу	01.12.2024	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	15.12.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 82 с., 30 рис., 7 табл., 51 джерело.

Тема: Моделі, методи та алгоритми побудови журналу подій для хмарних систем

Об'єкт дослідження: процеси обробки та аналізу журналів подій у хмарних системах.

Мета роботи: розробка та обґрунтування методів і підходів для видобування та обробки даних із журналів подій хмарних систем, які забезпечують ефективний інтелектуальний аналіз процесів і можливість вдосконалення управління бізнес-процесами.

Предмет дослідження: методи та алгоритми видобування й обробки даних із журналів подій для забезпечення інтелектуального аналізу процесів у хмарних системах.

Результати дослідження

В роботі рапропоновано модифіковану методику інтелектуального аналізу процесів для хмарних систем, яка враховує специфіку великих обсягів неоднорідних даних.

Висновок

Створено алгоритми вилучення, трансформації та аналізу подій для хмарних систем, що сприяють оптимізації бізнес-процесів. Отримані результати можуть бути корисними для розробників систем обробки подій у хмарних середовищах, а також для спеціалістів у галузі управління IT-процесами.

ХМАРНІ СИСТЕМИ, ОБРОБКА ЖУРНАЛІВ ПОДІЙ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ПРОЦЕСІВ, АРТЕФАКТНО-ОРІЄНТОВАНИЙ ПІДХІД, ВИДОБУВАННЯ ДАНИХ, БІЗНЕС-ПРОЦЕСИ

ABSTRACT

Master Thesis: 82 pp., 30 fig., 7 tab., 51 sources.

Thesis Subject: Models, methods and algorithms for building an event log for cloud systems

Research object: processes of processing and analysis of event logs in cloud systems.

The purpose of the work: development and substantiation of methods and approaches for extracting and processing data from event logs of cloud systems, which provide effective intellectual analysis of processes and the possibility of improving business process management.

Research subject: methods and algorithms of data extraction and processing from event logs to provide intelligent analysis of processes in cloud systems.

Research results

The paper proposes a modified method of intelligent analysis of processes for cloud systems, which takes into account the specifics of large volumes of heterogeneous data.

Conclusion

Algorithms for extracting, transforming and analyzing events for cloud systems have been created, contributing to the optimization of business processes. The obtained results can be useful for developers of event processing systems in cloud environments, as well as for specialists in the field of IT process management.

CLOUD SYSTEMS, EVENT LOG PROCESSING, INTELLIGENT PROCESS ANALYSIS, ARTIFACT-ORIENTED APPROACH, DATA MINING, BUSINESS PROCESSES

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ERP- Enterprise Resource Planning

CDN - Content Delivery Network

CI/CD - Continuous Integration/Continuous Delivery

DNS - Domain Name System

GUI - Graphical User Interface

IaaS - Infrastructure as a Service

K8s - Kubernetes

LDAP - Lightweight Directory Access Protocol

PaaS - Platform as a Service

REST - Representational State Transfer

SaaS - Software as a Service

SLA - Service Level Agreement

SSH - Secure Shell

SSL - Secure Sockets Layer

TCP - Transmission Control Protocol

ВСТУП

Актуальність теми.

З ростом обсягів даних, що генеруються у сучасних хмарних середовищах, виникає потреба в ефективному управлінні, обробці та аналізі цих даних для забезпечення стабільності і високої продуктивності IT-систем. Журнали подій у хмарних системах містять цінну інформацію про виконання бізнес-процесів, стан сервісів та взаємодію компонентів систем, однак для отримання інсайтів з цієї інформації потрібно застосовувати сучасні методи інтелектуального аналізу процесів (Process Mining). Зростання популярності хмарних технологій спричиняє збільшення складності та обсягів таких журналів, що потребує нових рішень для обробки та видобування подій.

Традиційні підходи до обробки журналів подій виявляються малоефективними для роботи з великими, неоднорідними даними, особливо у хмарних середовищах, де події часто є складними та взаємозалежними. У цьому контексті артефактно-орієнтований підхід дозволяє забезпечити гнучкість і точність обробки інформації, надаючи структуроване представлення процесів навіть у великих, розподілених системах. Це є особливо важливим для таких галузей, як фінанси, телекомунікації та електронна комерція, де від надійності та ефективності обробки подій залежать ключові аспекти бізнесу, як-от обслуговування клієнтів, управління ризиками та дотримання нормативних вимог.

Актуальність даної теми також підкреслюється швидким розвитком технологій великих даних та штучного інтелекту, які відкривають нові можливості для глибшого аналізу та оптимізації бізнес-процесів у хмарних середовищах. Поєднання цих технологій з інтелектуальним аналізом процесів дозволяє забезпечити не тільки збір і моніторинг подій, але й отримання нових знань про поведінку системи та користувачів, що є ключовим фактором для прийняття стратегічних рішень та підвищення конкурентоспроможності.

Отже, розробка методів і алгоритмів для обробки журналів подій у хмарних системах є актуальною науковою задачею, вирішення якої дозволить значно підвищити ефективність управління бізнес-процесами, забезпечити їхню стабільність і адаптивність в умовах масштабованих хмарних інфраструктур.

Мета дослідження – розробка та обґрунтування методів і підходів для видобування та обробки даних із журналів подій хмарних систем, які забезпечують ефективний інтелектуальний аналіз процесів і можливість вдосконалення управління бізнес-процесами.

Об'єкт дослідження – процеси обробки та аналізу журналів подій у хмарних системах.

Предмет дослідження – методи та алгоритми видобування й обробки даних із журналів подій для забезпечення інтелектуального аналізу процесів у хмарних системах.

Задачі дослідження:

- Провести огляд предметної області та дослідити особливості обробки неоднорідних даних журналів подій у хмарних середовищах.

- Визначити вимоги до побудови системи обробки журналів подій для хмарних систем.

- Розробити методику видобування та обробки подій із використанням артефактно-орієнтованого підходу.

- Реалізувати та протестувати розроблені методи й алгоритми, що забезпечують ефективний аналіз подій.

- Оцінити ефективність запропонованих методів для вдосконалення управління бізнес-процесами у хмарних системах.

Методи дослідження

Для досягнення мети дослідження використовувалися методи інтелектуального аналізу процесів (Process Mining), методи структурування та нормалізації даних, методи обробки подій у хмарних середовищах, а також методи реплікації та синхронізації баз даних.

Наукова новизна отриманих результатів

Розроблено та обґрунтовано новий підхід до обробки журналів подій у хмарних системах на основі артефактно-орієнтованої методики, що забезпечує підвищену точність у структурованому представленні подій.

Практичне значення результатів

Запропоновані методи й алгоритми можуть бути застосовані для розробки систем моніторингу та аналітики у хмарних середовищах, що дозволяє компаніям вдосконалювати управління бізнес-процесами, підвищувати продуктивність та забезпечувати стабільність хмарних інфраструктур.

Структура магістерської роботи. Робота складається зі вступу, трьох розділів та висновків. Загальний обсяг роботи становить 82 сторінки, і містить 30 рисунків, 7 таблиць, список використаних джерел із 51 найменування.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ОБРОБКИ НЕОДНОРІДНИХ ДАНИХ ЖУРНАЛІВ ПОДІЙ В ХМАРНИХ СИСТЕМАХ

1.1. Представлення області дослідження

Інтелектуальний аналіз процесів (Process Mining) - це швидкозростаюча дисципліна, кінцевою метою якої є покращення бізнес-процесів для підприємств. Він може бути описаний як трифазна процедура, що складається з вилучення інформації, перетворення інформації та отримання результатів і аналітичних висновків за допомогою різних методів інтелектуального аналізу процесів.

В рамках цього проекту проведено повний аналіз процесів хмарних систем, що охоплює всі три фази. Особлива увага приділяється перетворенню інформації, оскільки традиційні методи є трудомісткими та вимагають значного технічного досвіду, такого як знання SQL-скриптів.

Process Mining — це дисципліна, яка швидко розвивається, і її можна вважати сполучною ланкою між бізнес-процесами та бізнес-аналітикою. Кінцевою метою цієї галузі є вдосконалення бізнес-процесів підприємств. Щоб досягти цієї мети, потрібно виконати кілька необхідних кроків, які мають різні рівні складності. Зберігання даних, передача даних, перетворення даних і аналіз даних однаково важливі, коли йдеться про вдосконалення бізнес-процесів. Однак велика частина дослідницьких зусиль була витрачена на аналіз даних, без особливих зусиль, застосованих для перетворення даних. У цій галузі перетворення даних відноситься до процедури перетворення необроблених даних, що зберігаються в інформаційних системах, у чітко визначений формат, який називається журналом подій. В [3] автор визнав ключову роль журналів подій у видобутку процесів, заявивши, що «видобуток процесів стоїть чи падає з доступністю журналів подій». Без журналів подій процес інтелектуального аналізу не може бути виконаний

належним чином, оскільки вони служать відправною точкою аналізу процесів. Створення журналів подій не є тривіальним завданням, оскільки їхня якість значною мірою залежить від базових структур систем, що містять інформацію. Складність зазначено в Маніфесті процесу видобутку. Цей документ є публічною декларацією принципів і намірів, розроблених членами та прихильниками IEEE Task Force on Process Mining [4].

Він визначає п’ять рівнів зрілості журналів подій. Ці рівні зрілості дуже допомагають зрозуміти складність створення журналів подій. Найнижчий рівень зрілості стосується сценарію, коли інформація зберігається вручну несистематичним способом. Найвищий рівень зрілості пов’язаний з інформаційними системами, де всі дії та атрибути автоматично зберігаються з чіткою семантикою. Оскільки неможливо передбачити, чи буде коли-небудь досягнутий найвищий рівень зрілості у високому масштабі, необхідно докласти більше зусиль до етапу трансформації. Цього можна досягти шляхом вдосконалення методів перетворення струму. Необхідно оцінити весь підхід до створення журналу подій, а не лише кінцевий результат.

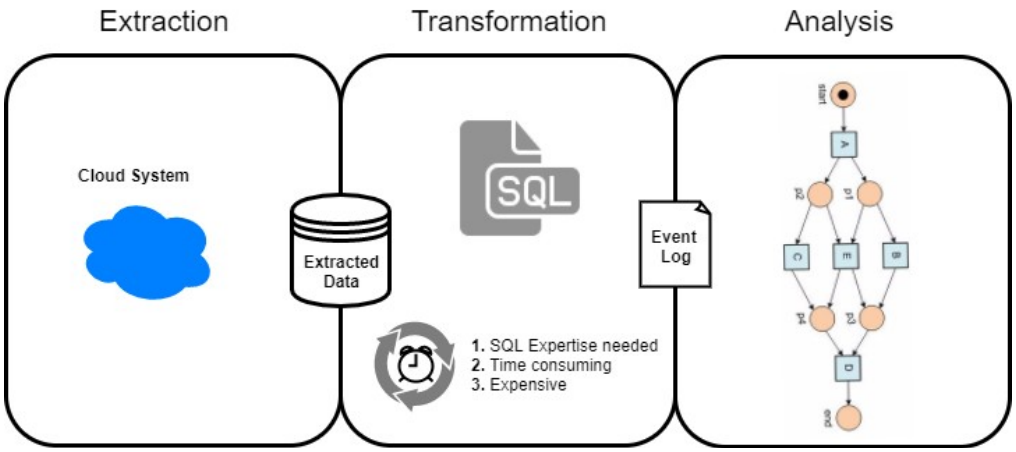


Рис. 1.1. Візуальне представлення етапів процесів видобутку і трансформації даних в журналах подій (логах)

Для створення ефективного рішення необхідно враховувати такі елементи, як проектування інформаційних систем, знання предметної

області, час обчислень і автоматизація. Традиційно журнали подій витягуються шляхом створення запитів SQL вручну [25]. Візуальне представлення етапів видобутку процесу можна побачити на рисунку 1.1.

Існує багато джерел, які пояснюють, як отримати журнали подій із систем ERP [13, 25, 37]. На ці приклади впливають різні виклики. Підходи, використані в цих джерелах, можуть отримати журнали подій із компромісами, що вимагають великої кількості ручних зусиль, знання сценаріїв SQL або знання домену. Всі ці прийоми можна віднести до традиційних підходів. В [34] автор запропонував техніку під назвою «підхід, орієнтований на артефакти», який може автоматизувати багато кроків у процесі. Крім того, він може обробляти сценарії SQL на внутрішньому рівні з меншою кількістю взаємодії з користувачем, ніж традиційні підходи. Тим не менш, ця техніка була каталогізована кінцевим користувачем як складна і рідко використовується. У цій роботі було створено прототип, щоб продемонструвати, як цю техніку можна використовувати як основу для створення журналів подій. Прототип здатний усунути деякі обмеження традиційних підходів, такі як знання SQL і час, витрачений на автоматизацію різних частин процедури.

Оскільки попередні дослідження були зосереджені виключно на локальних системах, таких як SAP і Oracle, цей проект має інший шлях. Хмарні системи використовують швидкі темпи, оскільки все більше компаній вирішують перейти до «цифрового світу». Хмарні обчислення та програмне забезпечення як послуга (SaaS) використовують переваги можливості спільно використовувати інфраструктуру та програмне забезпечення між кількома організаціями. Кінцевим результатом є скорочення часу налаштування, оплата за використання, зниження цін і зменшення зусиль на обслуговування та управління. Незважаючи на нинішній розвиток хмарних систем і різноманітні переваги, які вони надають, у цій галузі не так багато досліджень процесів. З цієї причини дві популярні

хмарні системи (Salesforce і Jira) були обрані як цільові вихідні системи для поточного дослідження.

1.2. Опис використовуваного підходу

Метою даного розділу є огляд обраної процедури для проведення наскрізного аналізу хмарних систем. Від кінця до кінця розуміється кожен крок, пов'язаний з отриманням інформації та висновків за допомогою методів аналізу процесів. Перший крок полягає у формуванні загальних вимог. Після визначення загальних вимог було проведено аналіз існуючих інструментів, у результаті чого було обрано орієнтований на артефакти підхід як основну технологію для створення журналів подій. Пропонований підхід аналізу представлено на рисунку 1.2.

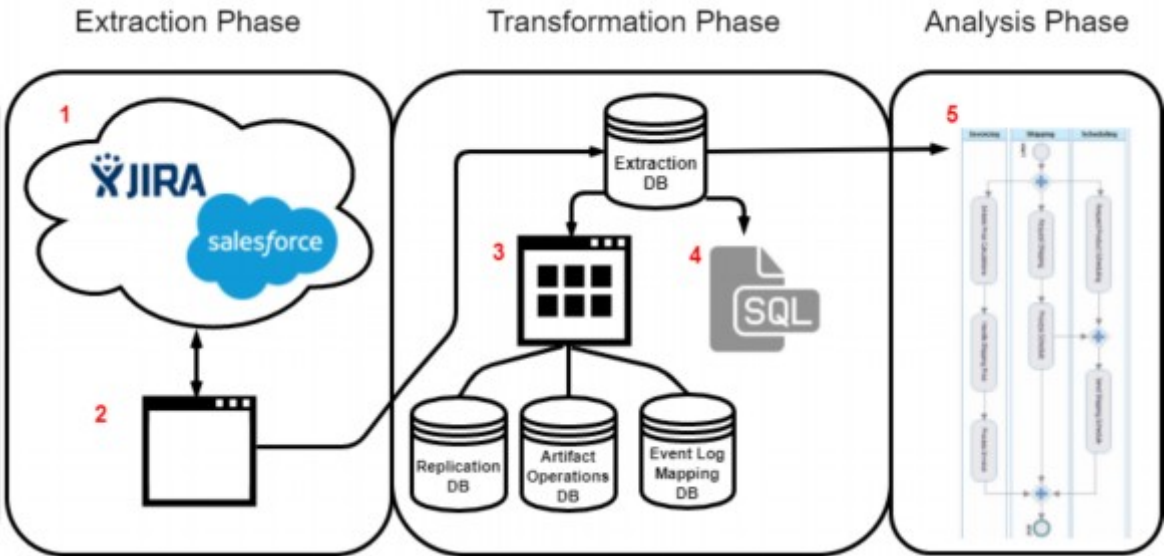


Рис. 1.2. Підхід аналізу даних журналу подій

На рисунку 1.2 показано 1 - хмарні системи, 2 - API Extractor, 3 – пропонований екстрактор, 4 - сценарії SQL, 5 - методи аналізу процесу обробки даних.

Наступним кроком є перетворення загальних вимог у дуже конкретні характеристики. Ці функції використовуватимуться для розуміння того, що

вже розроблено та що потрібно інтегрувати в пропонований екстрактор або в будь-який інший компонент технічної настройки. Після виконання конкретних вимог буде показано покроковий посібник із трьох етапів інтелектуального аналізу процесу для Jira та Salesforce.

Усі результати запропонованої процедури остаточно впроваджуються у допоміжне програмне забезпечення. Усі кроки етапу трансформації реалізовано як інструмент екстрактор з відкритим кодом.

1.3. Особливості інтелектуального аналізу процесів (Process Mining)

Process Mining (інтелектуальний аналіз процесів) - це метод аналізу бізнес-процесів, який використовує дані з IT-систем компанії, щоб зрозуміти, як насправді відбуваються процеси. Це дозволяє виявити вузькі місця, неефективність та можливості для покращення.

Етапи Process Mining наступні:

- Збір даних: Process Mining використовує дані з журналів подій різних IT-систем, таких як CRM, ERP, BPM, бази даних тощо. Ці дані містять інформацію про кожен крок у процесі, хто його виконав, коли і скільки часу це зайняло.

- Візуалізація: Дані перетворюються на візуальні моделі процесів, які легко зрозуміти. Це можуть бути діаграми, графіки, теплові карти тощо.

- Аналіз: За допомогою спеціальних алгоритмів та технік аналізуються моделі процесів, щоб виявити відхилення від стандартів, вузькі місця, неефективність та інші проблеми.

- Покращення: На основі отриманих результатів розробляються рекомендації щодо оптимізації процесів, підвищення ефективності та зниження витрат.

Process Mining - це потужний інструмент для аналізу та покращення бізнес-процесів, який допомагає компаніям підвищити ефективність, знизити витрати та досягти кращих результатів.

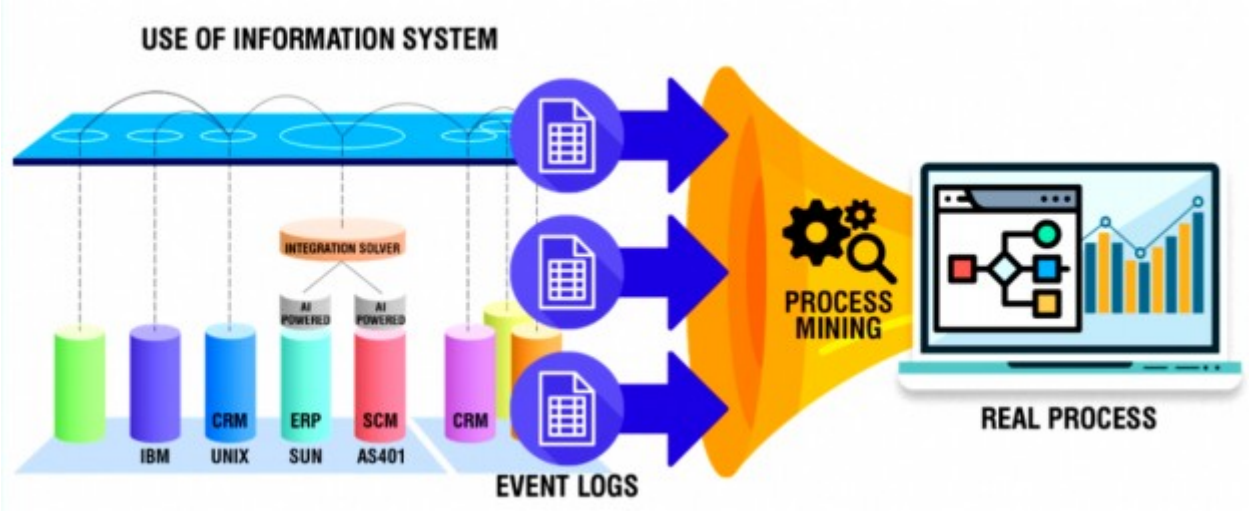


Рис. 1.3. Візуальне представлення Process Mining

Інтелектуальний аналіз процесів можна визначити як міст, який з'єднує традиційний аналіз процесів на основі моделі (наприклад, моделювання та інші методи управління бізнес-процесами) і методи аналізу, орієнтовані на дані, такі як машинне навчання та інтелектуальний аналіз даних [3]. Загальна ідея технології полягає у виявленні та вилученні цифрових слідів , що зберігаються в інформаційних системах. Цифрові сліди - це дії, які були виконані в даній системі. Доступність цих дій пов'язана зі складністю інформаційної системи. Проте потенційно можливо відстежувати такі дії, як клієнти, які купують товари на веб-сайті, банкомати, які приймають зняття готівки, або редагують ціну транзакції замовлення на покупку в системі CRM. Коли ця інформація витягнута, очищена та перетворена у формат, званий журналами подій, можна виявити модель, що описує поведінку процесу. Існує багато досліджень на цю тему, з великою кількістю публікацій про відкриття процесів. Справжня перевага інтелектуального аналізу процесів з'являється після виявлення моделі. Такі методи, як перевірка відповідності [4, 16] і соціальні взаємодії [24] дають потужну інформацію про що відбувається в бізнес-процесах. Крім того, можна легко побачити час пропускної здатності між діями, виявивши вузькі місця та порушення процесу. Усі ці методи мають можливість аналізувати велику кількість даних

одночасно, що робить їх набагато ефективнішими, ніж старі методи вдосконалення процесів, засновані на інтерв'ю з користувачами або віртуальні процеси, які не були створені з реальними даними.

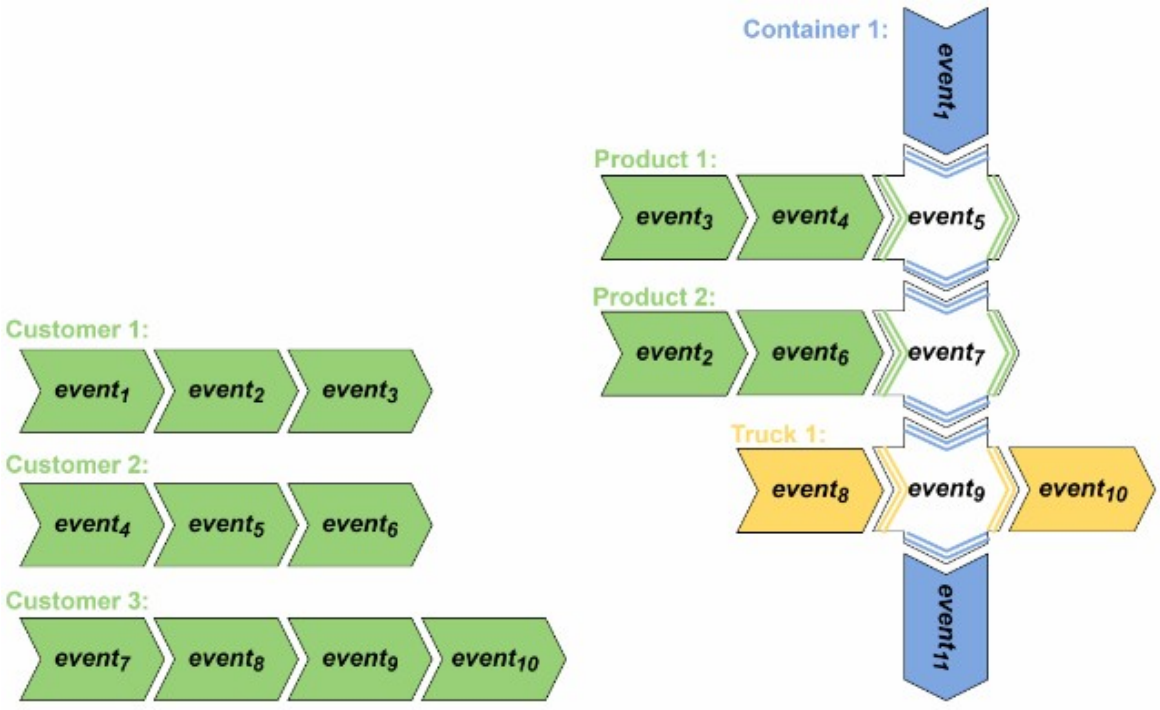
Оскільки інформаційні системи розвиваються та стають надійнішими, кількість цифрових слідів зростає. Це покращує набори даних і відкриває двері для аналізу інтелектуального аналізу процесів, але також ускладнює вилучення, очищення та перетворення даних. Створення журналів подій є складним завданням, яке обговорювалося в багатьох дослідженнях [26, 27, 35]. На рисунку 1.4 подано фрагмент журналу подій.

_CASE_KEY	ACTIVITY_EN	EVENTTIME
50024000001A7SAAA0	Create Case	2015-06-07 18:16:00.000
50024000001A7SAAA0	Change Status to Closed	2015-06-07 21:35:50.000
50024000001A7SAAA0	Change Status to Approved	2015-06-07 21:50:31.000
50024000001A7SAAA0	Change Status to Closed	2015-06-17 13:09:24.000
50024000001A7SjAAK	Create Case	2015-06-07 18:20:47.000
50024000001A7SjAAK	Change Status to Closed	2015-06-07 21:35:42.000
50024000001A7SjAAK	Change Status to Approved	2015-06-17 13:05:25.000
50024000001A7SjAAK	Change Status to Closed	2015-06-17 13:09:30.000
50024000001A7TIAA0	Create Case	2015-06-07 18:25:15.000
50024000001A7TIAA0	Change Description	2015-06-07 18:30:15.000
50024000001A7TIAA0	Change Status to Closed	2015-06-07 21:35:34.000

Рис. 1.4. Приклад журналу подій

1.4. Традиційний підхід обробки даних журналу подій

Традиційний підхід можна описати як класичний метод отримання журналів подій. Зазвичай інформація зберігається в реляційній базі даних, оскільки більшість інформаційних систем використовують цей тип бази даних. Отже, використовується підхід сценаріїв SQL. Перший крок полягає в розумінні структури реляційної бази даних. Необхідно ідентифікувати відповідні таблиці та взаємодію у формі зв'язків зовнішнього ключа.



а) традиційний журнал подій б) об’єктно-орієнтований журнал подій

Рис. 1.5. Типи журналів подій

На рисунку 1.5 традиційні журнали подій мають вигляд а), де кожна подія пов'язана рівно з одним об'єктом, який називається кейсом. Об'єктно-орієнтовані журнали подій б) відмовляються від цього обмеження, події можуть бути пов'язані з кількома об'єктами різних типів.

Наступним кроком є вибір ідентифікатора справи. Ідентифікатор випадку або екземпляр процесу є первинним ключем однієї з таблиць у системі. Це ключове рішення, оскільки воно визначатиме сутність або об’єкт для аналізу. Залежно від процесу вибір ідентифікатора справи може бути комбінацією таблиць. Крім того, існують процеси, у яких ідентифікатор справи є дуже інтуїтивно зрозумілим і його вибір не складний. Останнім кроком методу перед написанням сценаріїв SQL є ідентифікація дій. Діяльність ідентифікується мітками часу. Зазвичай існує таблиця журналу змін, яка містить усі зміни в системі.

Щоб виявити дії, потрібен запит на приєднання таблиці ідентифікаторів справ до таблиці, яка містить інформацію про дії. Залежно від складності системи ці операції можуть бути складнішими. Результат цих запитів

зазвичай складається з назви дії та позначки часу дії. Однак бувають випадки, коли до журналу подій додаються додаткові атрибути. У таких випадках у запитах потрібно спеціально вибрати додаткові атрибути.

Описана процедура вимагає великих зусиль, оскільки ці запити потрібно писати вручну. Крім того, необхідні знання предметної області, щоб зрозуміти, які дії та атрибути мають відношення до конкретного процесу. Остаточний журнал подій є продуктом багатьох ітерацій із залученням різних зацікавлених сторін, таких як експерти з SQL (науковці даних або бізнес-аналітики), адміністратори баз даних і власник бізнесу.

1.5. Артефактно-орієнтований підхід побудови журналу подій

У цьому розділі ми досліджуємо вибрану техніку вирішення проблеми створення журналів подій. Ми починаємо цей розділ, розглядаючи походження процедури, перш ніж проходити кожен із кроків. Численні проблеми, що впливають на методологію, будуть виявлені в поєднанні з нещодавньою роботою в цій області. Причини використання артефакто-орієнтовано підходу будуть викладені в кінці розділу.

Як можна зробити висновок з назви, цей підхід використовує артефакти. Артефакт — це конкретна, ідентифікована, самоописувана частина інформації, яка може бути використана бізнесменом для справжнього ведення бізнесу. Артефакти поєднують як аспекти даних, так і аспекти процесу в цілісну одиницю та служать основними будівельними блоками, з яких будуються моделі бізнес-операцій і процесів. Концепція артефакту йде рука об руку з життєвим циклом артефакту. Обробка артефактів – це спосіб описати діяльність підприємства. Життєвий цикл артефакту охоплює наскрізну обробку конкретного артефакту, від створення до завершення та архівування.

В [12] автор створив автоматичну процедуру, яка поєднала ці концепції з кількома ідеями з різних сфер. Метою цієї процедури було вилучення

журналів подій з реляційних баз даних. Він розробив прототип під назвою XTract,. На додаток до артефактів і виявлення життєвого циклу артефактів, він розробив і реалізував кілька кроків, щоб увімкнути автоматичне генерування журналів подій. Рисунок 1.6 відображає графічне представлення всіх кроків.

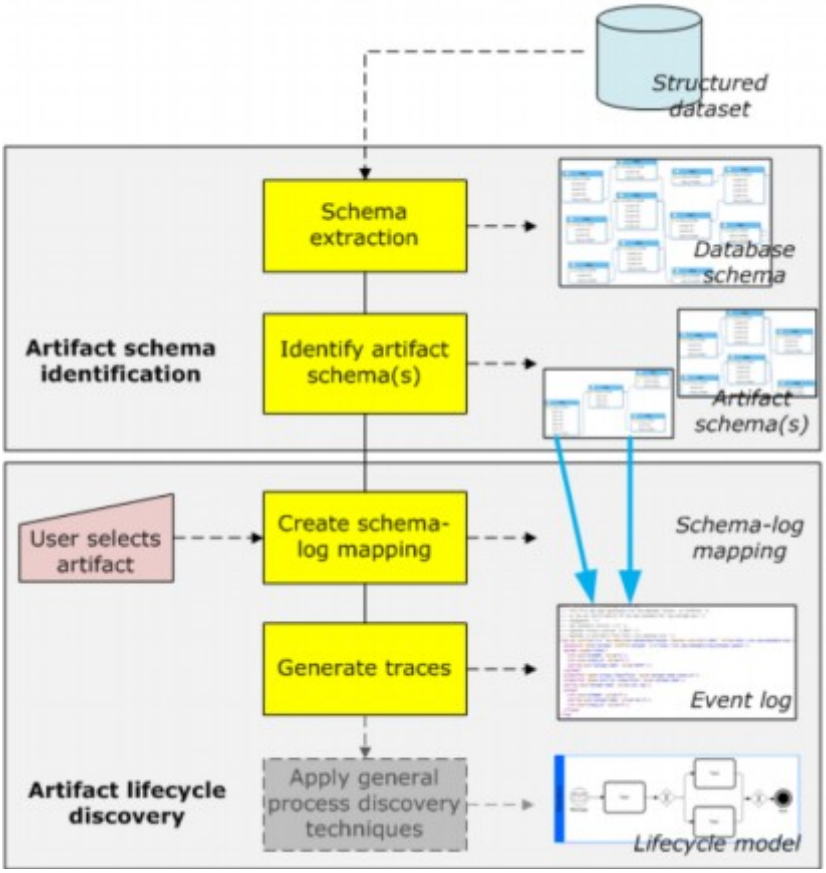


Рис. 1.6. Оригінальний метод артефакто-орієнтованого підходу

Вилучення схеми

Вилучення схеми визначається тут як вилучення структурної інформації (наприклад, ключі кандидатів і зв'язки між сутностями) зі структурованих даних (наприклад, таблиць). Він приймає як вхідні дані набір структурованих (табличних) даних і створює:

1) первинні ключі для кожної таблиці,

2) домен для кожного елемента (стовпця) у кожній таблиці,

3) зв'язки (зовнішні ключі) між таблицями .

Визначення схеми артефактів

На цьому кроці визначається підмножина набору даних, яка представляє інтерес для кожного конкретного артефакту. Ідея полягає в тому, щоб розділити повну схему на кілька кластерів (по одному для кожного артефакту) і призначити кожну таблицю одному або декільком кластерам. Крім того, для кожного кластера буде обрано репрезентативну головну таблицю. Ця головна таблиця містить інформацію про екземпляр для конкретного артефакту: кожен екземпляр артефакту можна ідентифікувати за первинним ключем головної таблиці.

Створення зіставлення схеми з журналом

На цьому етапі створюється зіставлення між схемою артефакту та отриманим журналом подій. Він приймає як вхідні дані:

1) схему артефакту,

2) структурований набір даних, описаний схемою артефакту, і створює набір типів подій, визначених у наборі даних, і відображення набору даних у ці типи подій.

Відображення, знайдене таким чином, описує, як витягти різні події з набору даних. Зверніть увагу, що в цьому випадку мета журналу подій полягає в тому, щоб використовувати його для виявлення життєвого циклу артефакту за допомогою методів аналізу процесу. Оскільки це залежить від мети проекту інтелектуального аналізу процесів, що включати в журнал подій це слід взяти до уваги.

Генерація слідів

Використовуючи зіставлення, знайдене на попередньому кроці, події можуть бути згенеровані з набору даних. Підхід, описаний у [9], був розроблений для подібної мети: він приймає як вхідні дані набір даних і відображення та створює журнал подій. Подібним чином для кожного артефакту цей крок має приймати як вхідні дані набір даних і відображення для артефакту, як описано в попередньому розділі. Результатом має бути журнал подій у форматі eXtensible Event Stream (XES). Було обрано формат XES, оскільки це єдиний стандартизований формат журналу подій. Він був

розроблений для обміну даними журналу подій у простому, виразному та гнучкому форматі, одночасно дозволяючи розширення формату прозорим способом [13]. Для наших технічних налаштувань ми не зацікавлені у створенні виводу у форматі XES.

Застосування методів виявлення процесів

Метою виявлення життєвого циклу артефакту є виявлення як внутрішнього життєвого циклу артефакту , так і його взаємодії з іншими артефактами, таким чином повністю описуючи, як працює артефакт. За допомогою журналу подій, створеного на попередньому кроці, можна використовувати різноманітні методи виявлення процесу для створення життєвого циклу для кожного артефакту.

В [26] визначено кілька проблем щодо екстратктора. Ці проблеми можна підсумувати як:

1) Неможливо визначити складні зовнішні ключі,

2) Неможливо ідентифікувати кілька артефактів в одній таблиці та кілька типів подій в одному стовпці,

3) Проблеми з продуктивністю, пов’язані з вилученням журналів,

4) Відсутність взаємодії між артефактами.

Ці проблеми вирішуються шляхом створення взаємодії між артефактами. Оскільки кожен артефакт створює журнал подій, артефакт можна вважати процесом. Отже, взаємодії артефактів спрямовані на виявлення дуже конкретних артефактів, а потім на пошук зв’язку в межах діяльності. Хоча ця нова процедура здатна вирішити більшість проблем, вона породжує дві проблеми. Найвідомішим є те, що немає чіткого способу візуалізації цих артефактів та їх взаємодії. По-друге, визначення взаємодії між артефактами має виконуватися кінцевим користувачем вручну, що призводить до складного та неавтоматичного методу. Рисунок 1.7 відображає графічне представлення кількох взаємодіючих артефактів із високорівневого подання. На додаток до цих проблем, прототип, створений автором (XTract v2), є приватним, а вихідний код недоступний.

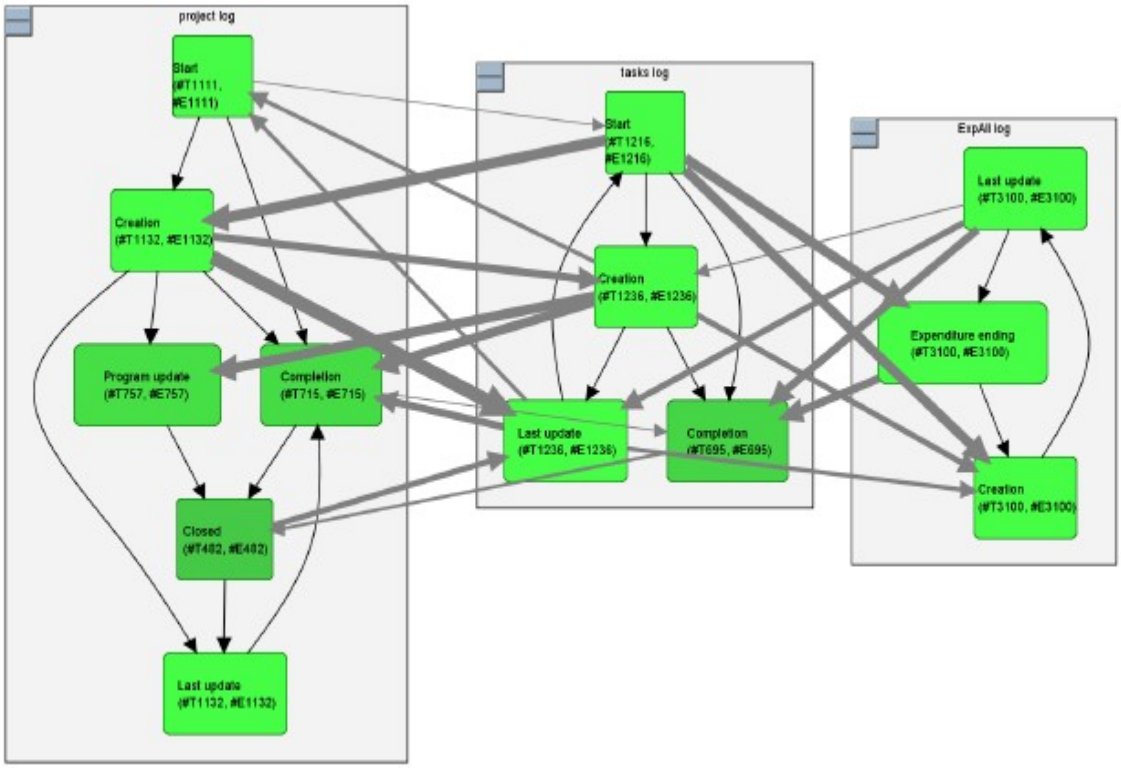


Рис. 1.7. Візуалізація взаємодіючих артефактів

Незважаючи на те, що точну роботу з [26] не можна використати повторно, багато знахідок і аналіз XTract v1 можна використовувати як орієнтир для вирішення заявлених відомих проблем. Оскільки основні цілі поточного проекту йдуть рука об руку з економією часу, зменшенням кількості складності та створенням правильних журналів подій на фазі трансформації, артефакто-орієнтований підхід виглядає ефективною технологією для архівування цих цілей.

1.6. Опис існуючих підходів обробки даних журналу подій

У поточному розділі розглядаються різні технології, які вивчалися перед вибором.

1.6.1. Платформа Xesame

Xesame (раніше називався XESma або XES Mapper) був створений JCAM Buijs. Мета цього прототипу полягала в тому, щоб створити програмне

забезпечення, яке дозволило б відображати бази даних інформаційних систем у файлах XES. Деякі частини використовуються в XTract v1. XTract додає додаткові можливості, такі як зіставлення схеми з журналом, що покращує автоматизацію. Хоча Xesame не використовується безпосередньо, концепції та спостереження, розроблені у згаданій роботі, використовуються опосередковано.

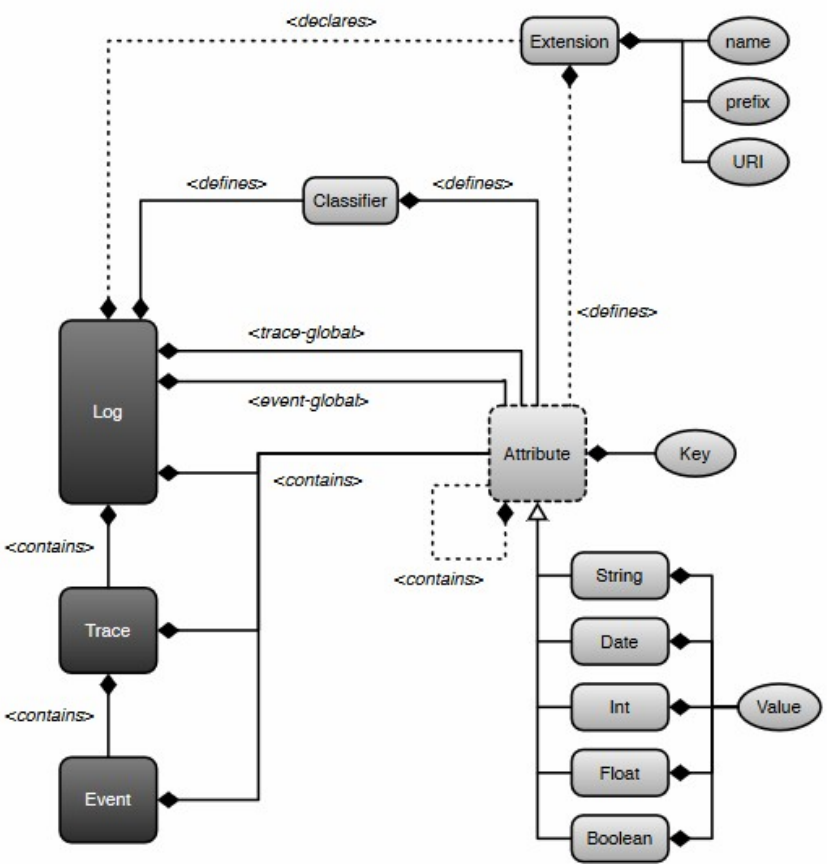


Рис. 1.8. XES мета модель

1.6.2. Метамодель OpenSlex

OpenSlex — це метамодель, представлена в [18]. Ця метамодель надає універсальну техніку, яку можна використовувати для розділення аналізу та вилучення даних. Концепція моделі даних поверх інформаційних систем надає нові функції, такі як простий спосіб зміни ракурсів у журналі подій. Однак існує дуже фундаментальне обмеження: для того, щоб перетворити дані для введення їх у модель даних OpenSlex, перетворення даних з різних систем має бути виконане одним. Отже, ця техніка не стосується цілей поточного проекту.

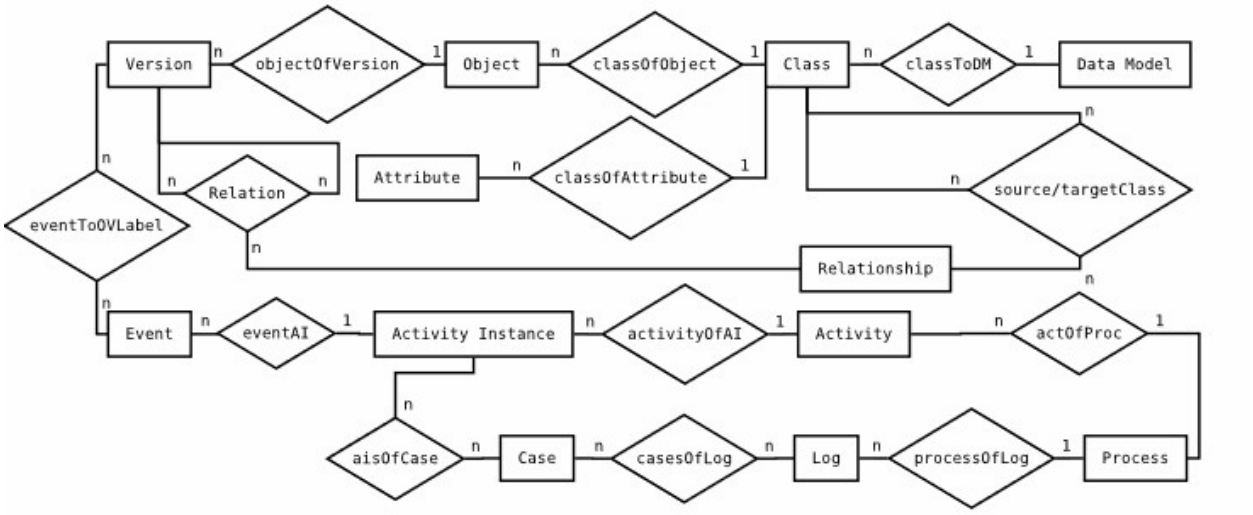


Рис. 1.9. ER-діаграма мета моделі OpenSlex

1.6.3. Підхід на основі онтології

Процедура на основі онтології була запропонована для вилучення журналів подій із систем реляційних баз даних [10]. Хоча доведено, що метод генерує журнали подій у формі файлів XES, існує кілька обмежень. З одного боку, поняття, пов'язані з подіями, повинні бути визначені вручну фахівцями в онтології домену. З іншого боку, користувачам потрібна експертиза SPARQL для написання анотацій у системі. Тому цю методику відкидають.

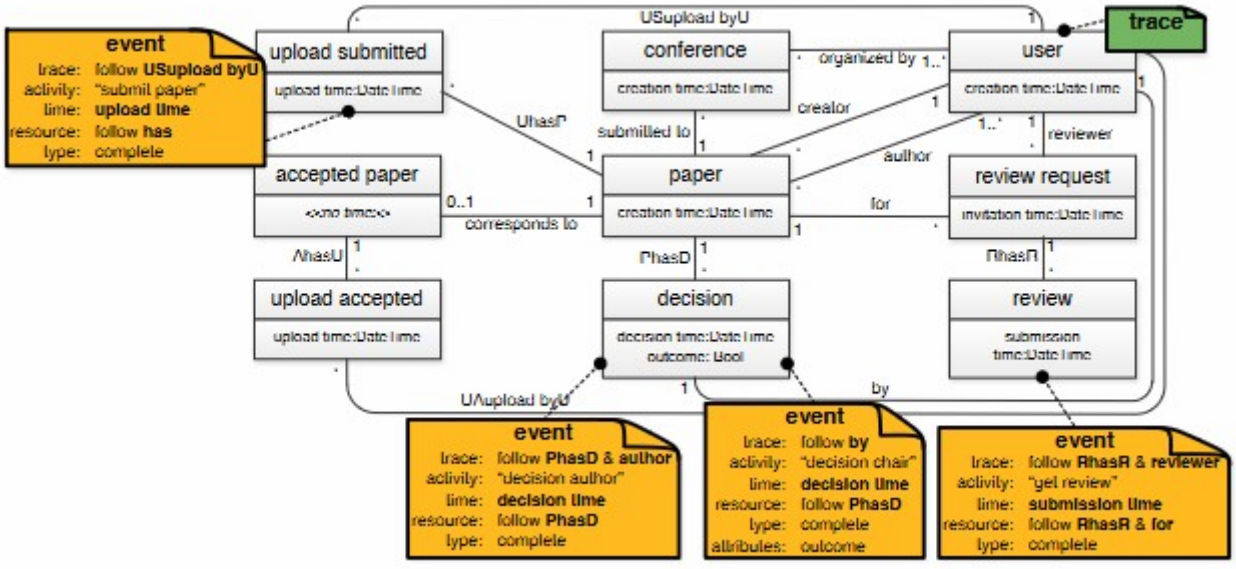


Рис. 1.10. Анотації онтології домену для відстеження динаміки користувачів

На відміну від попередніх робіт, вилучення керується концептуальним представленням предметної області, що цікавить, у вигляді онтології. З одного боку, ця онтологія пов'язана з базовими застарілими даними, використовуючи добре зарекомендовану парадигму доступу до даних на основі онтології (OBDA).

З іншого боку, наша платформа дозволяє збагачувати онтологію за допомогою орієнтованих на користувача анотацій вилучення журналів, які можна гнучко використовувати для забезпечення різних орієнтованих на журнал подань даних. Потім розробляються різні режими доступу до даних, щоб переглядати застарілі дані крізь призму XES.

Висновки до розділу

В даному розділі проведено огляд предметної області обробки неоднорідних даних журналів подій у хмарних системах, що є основою для подальших досліджень та аналізу. Окреслено важливість аналізу подій у хмарних системах, де великі обсяги різнорідних даних потребують спеціалізованих підходів для ефективної обробки та інтелектуального аналізу. Описано кілька підходів до обробки, включаючи традиційний метод, артефактно-орієнтований підхід та методи, що базуються на концепціях Process Mining (інтелектуальний аналіз процесів). Процесний підхід надає можливість глибшого розуміння операцій у хмарних системах і підтримує оптимізацію процесів.

Розглянуто інструменти Xesame, метамодель OpenSlex та підхід на основі онтології. Кожен із них має свої особливості щодо представлення, обробки та аналізу подій. Xesame забезпечує гнучке перетворення даних, OpenSlex дозволяє створювати метамоделі для покращеної інтеграції, а підхід на основі онтологій забезпечує семантичну сумісність. У дослідженні порівняно різні методи та інструменти, що дозволяють покращити точність та швидкість обробки подій, особливо в умовах хмарних середовищ, де

велика кількість подій створює високі вимоги до продуктивності та масштабованості.

Загалом, розділ дає комплексний огляд сучасних методів обробки даних журналів подій, що забезпечує основу для вибору найбільш придатного підходу для конкретних завдань у хмарних системах.

РОЗДІЛ 2. МЕТОДИ, МЕТОДИКА ТА ВИМОГИ ДО СИСТЕМ ОБРОБКИ ТА ВИДОБУВАННЯ ДАНИХ В ЖУРНАЛАХ ПОДІЙ

2.1. Процес інтелектуального аналізу процесів для хмарних систем

Хмарні обчислення визначаються як модель для забезпечення повсюдного, зручного мережевого доступу на вимогу до спільного пулу конфігурованих обчислювальних ресурсів (наприклад, мереж, серверів, сховищ, програм і служб), які можуть бути швидко надані та звільнені з мінімальними зусиллями адміністратора або взаємодії постачальника послуг [29].

Можливості хмарних обчислень щодо ефективності та доступності створили неймовірну кількість служб, таких як Amazon EC2/S3, Google Apps, Microsoft Azure. Ці служби забезпечують необхідну інфраструктуру для систем з різними спеціалізаціями, такими як CRM (SalesForce, SAP Ariba, SugarCRM, Microsoft Dynamics CRM), служби зберігання (Dropbox, Box), людські ресурси (SAP Success Factors, Workday, Cornerstone) серед багатьох інших областей. Існує термін «міжорганізаційний аналіз процесів», щоб скористатися перевагами хмарних обчислень і хмарних систем, щоб дозволити організаціям вчитися одна в одної та вдосконалювати свої процеси [42].

У дослідженні [19] автори застосували методи інтелектуального аналізу даних і процесів до веб-сервісів. Вони запропонували метод під назвою Web Services Interaction Mining (WSIM). Однак цей механізм не можна використовувати для наших цілей, оскільки немає визначеної структури, яку можна легко адаптувати у вигляді централізованого інструменту. Цей підхід витягує інформацію з різнорідних джерел і вимагає значних зусиль у розробці. Веб-скрапінг стикається з подібними проблемами.

Розглядаючи стандартні способи вилучення інформації, що зберігається в хмарних системах, найпоширеніша процедура використовує

інтерфейси прикладного програмування (API). Ці інтерфейси забезпечують механізм повторного використання коду, щоб програмісти могли створювати роботу, яку вже зробили інші програмісти (або вони самі), а не починати з нуля кожну програму. Крім того, часто потрібно використовувати API, оскільки низькорівневий доступ до системних ресурсів (таких як графіка, мережа, файлова система тощо) доступний лише через захищені API [32]. Тим не менш, API сильно залежать від рівня їхньої складності. Кожен власник системи несе відповідальність за документацію та обслуговування API протягом тривалого часу. Робіллард та інші вказали різні причини, чому ці інтерфейси важко вивчити, а документація може бути величезним чинником у зниженні продуктивності програмістів [40].

З огляду літератури можна зробити висновок, що програмні інтерфейси додатків є найкращим варіантом для вилучення даних із хмарних систем, і для кожної окремої системи необхідно розробити API Extractor. Однак розробка цього екстрактора може бути складною через такі фактори, як документація та закритий доступ до ключових даних. Щоб зрозуміти цю проблему, необхідно порівняти відмінності вилучення між хмарними системами та локальними системами. Якщо дивитися на найпопулярніші локальні системи (тобто SAP і Oracle), то існують інструменти, які обробляють вилучення інформації шляхом копіювання реляційних баз даних. Можна відфільтрувати витягнуті дані, вказавши такі параметри, як дата, але базова структура реляційної бази даних залишиться незмінною.

Друга проблема полягає в перетворенні видобутої інформації у відповідний журнал подій для аналізу процесів. Повний огляд цієї проблеми для систем ERP надано в [27].

В [33] надали повний огляд необхідних операцій для перетворення реляційних схем у моделі високого рівня. Високорівневі моделі можна визначити як ефективну структуру представлення даних для цілей аналізу процесу. Наступні операції з даними взяті зі згаданої роботи:

1) Горизонтальне розділення: спеціалізує загальну сутність на кілька різних таблиць залежно від їх типу. Наприклад, «Документи» поділяються на «Документи продажу» та «Документи поставки» з різними таблицями.

2) Вертикальне розділення: розподіляє властивості однієї сутності в декілька різних таблиць. Наприклад, «Зміни» до «Документа поставки» зберігаються не в таблиці «Документи поставки», а в окремій таблиці «Зміни документів».

3) Горизонтальне антирозбиття: узагальнює дані з кількох об’єктів в одну таблицю. Наприклад, усі зміни різних типів документів зберігаються в одній таблиці «Зміни документів», а не в окремих таблицях.

4) Вертикальне антирозбиття об’єднує атрибути кількох об’єктів в одну таблицю. Наприклад, «Документи про продаж» об’єднує атрибути для «Замовлення на продаж» і «Замовлення на повернення» (хоча «Довідковий ідентифікатор» потрібен лише для «Замовлення на повернення»). Приклади також показують, що одна таблиця може бути результатом кількох таких операцій.

Ці операції добре переносяться з попередньо вивчених систем (SAP і Oracle) у хмарні системи, особливо вертикальне антирозбиття.

Третя проблема пов'язана з рівнем деталізації в назвах діяльності. Деталізацію діяльності можна визначити як кількість деталей, виражену в назві події. Наприклад, коли відбувається зміна, назва такої дії може бути «Зміна X», де X — конкретна сутність, яка змінилася. Якщо доступні конкретні значення, пов’язані зі змінами, дії можна назвати «Зміна X на Y» або «Зміна X з Z на Y», де Z і Y — це попередні та нові значення після зміни. Під час цієї роботи ми використовуємо інформацію, що зберігається в таблицях змін, щоб визначити рівні деталізації.

Відкрита проблема в інтелектуальному аналізі процесів відома як дивергенція та конвергенція. У [9] обидва терміни визначені наступним чином:

Визначення 2.1. (конвергенція)

Одна і та сама дія виконується на кількох екземплярах процесу одночасно.

Визначення 2.2 (Розбіжність)

Для одного екземпляра процесу та сама дія виконується кілька разів.

В одному з наших прикладів згенерований журнал подій страждає від конвергенції. Тим не менш, це прийнято, оскільки діяльність, яка відбувається в кількох екземплярах процесу одночасно, надає більше інформації щодо реального процесу. Можливим рішенням було б ігнорувати діяльність, яка викликає проблему.

Останнє завдання полягає у застосуванні методів інтелектуального аналізу процесу (тобто виявлення моделі, перевірки відповідності, соціальних взаємодій тощо) до згенерованого журналу подій, для чого повинні використовуватись існуючі інструменти інтелектуального аналізу процесу.

2.2. Представлення вимог до побудови журналу подій для хмарних систем

Метою цього розділу є познайомити з вимогами до вилучення журналу подій, що висуваються інтелектуальним аналізом процесів як технологією. Ці елементи пов’язані з вилученням інформації з хмарних систем, моделюванням даних у таких системах і форматуванням журналів подій. На завершення в цій главі представлені вимоги високого рівня, які будуть використані для створення можливого рішення.

Вилучення даних із хмарних систем відбувається за стандартною процедурою, якщо ці системи забезпечують доступ до даних через API. Цей інтерфейс дозволяє витягувати інформацію структурованим способом. Тим не менш, тип інформації, обсяг і форматування сильно залежать від доступного API. Як наслідок, уся необхідна інформація може бути

недоступною, а документація може бути рідкісною. Також можливо, що існують різні версії API, оскільки компанії постійно оновлюють ці інтерфейси. До теперішнього часу розроблено понад шість різних версій, і деякі з цих версій більше не доступні . Оскільки різні системи обробляють API по-різному, ця робота не в змозі створити придатний для всіх програмний інструмент для вилучення даних хмарних систем. Вимога полягає в тому, щоб (REQ1) створити спеціальний програмний інструмент API, який отримує всю значущу інформацію, доступну з цільової хмарної системи. Спеціальний програмний інструмент API буде називатися API екстрактором.

Друга вимога випливає з базової природи технології процесу видобутку. Для того щоб застосувати численні методи інтелектуального аналізу процесів, розроблені з часом, потрібен журнал подій. Отже, (REQ2) потрібен спосіб перетворення вилучених даних у журнали подій .

Остання вимога з цього розділу полягає в додаванні атрибутів до таблиць діяльності. Журнали подій можуть містити більше інформації, наприклад, ресурс, який виконав дію. З цієї причини необхідна можливість (REQ3) додати додаткові атрибути до журналу подій.

Однією з головних цілей цього проекту є скорочення часу на етап трансформації. Як було пояснено в попередньому розділі, поточний метод здійснення перетворень полягає у створенні та запуску сценаріїв SQL. Використання цих сценаріїв має кілька обмежень. Потрібна особа, яка має знання SQL, і вона має створити сценарій SQL для кожної системи. Іноді система містить інформацію для кількох процесів, тому потрібно розробити кілька сценаріїв. Крім того, для розуміння кожної системи потрібен різний час. Метою має бути (REQ5) скорочення часу, необхідного для створення журналів подій.

Окрім скорочення часу, ще одна вимога пов’язана зі зменшенням технічної експертизи. Отже, (REQ6) слід надати рішення, яке дозволяє фахівцям, які не володіють SQL, створювати журнали подій.

Таблиця 2.1.

Представлення вимог високого рівня

ID	Requirement
REQ1	API extractor to obtain available information from a cloud system.
REQ2	Transform extracted data into activities table.
REQ3	Add additional attributes to the activities table.
REQ5	Reduce the amount of time needed to generate event logs.
REQ6	Solution which allows non-SQL experts to generate event logs.
REQ7	Only one event log can be used as input.
REQ10	Add sorting column to the activities table.

Залежно від хмарної системи, вона може містити інформацію для одного або кількох процесів. Незважаючи на те, що існують різні дослідження щодо створення журналів подій кількох процесів, більшість інструментів аналізу процесів приймають один журнал подій як вхідні дані. Таким чином, (REQ7) Лише один журнал подій можна використовувати як вхідні дані.

Нарешті, необхідно визначити порядок дій, коли вони мають однакову позначку часу. Цю проблему можна вирішити, додавши стовпець сортування (REQ10) у журнал подій. Рисунок 2.1 ілюструє, як виглядає сортування в журналі подій.

CASE_ID	ACTIVITY_NAME	EVENT_TIME	Sorting
5002400000e4fKcAAI	Create Case	2016-08-24 14:17:55.000	1
5002400000e4fKcAAI	Change Owner	2016-08-24 14:17:56.000	50
5002400000e4fKcAAI	Send or Receive Email	2016-08-24 14:17:56.000	120
5002400000e4fKcAAI	Change Owner	2016-10-11 09:30:00.000	50
5002400000e4fKcAAI	Create internal Comment	2016-12-20 09:16:12.000	30
5002400000e4fKcAAI	Change Status to Closed	2016-12-20 09:16:15.000	10

Рис. 2.1. Приклад атрибута сортування

2.3. Фази процесу майнінгу для хмарних систем

Метою цього розділу є представлення методу наскрізного аналізу процесів для аналізу хмарних систем і архітектуру для реалізації методу,

який задовольняє REQ1 - REQ10. Зокрема, для вирішення підпроблеми генерації журналу подій з реляційної бази даних ми пропонуємо дотримуватися артефакто-орієнтованого підходу. Перш ніж називати всі функції, важливо зрозуміти різні елементи дизайну. Для початку будуть представлені фази видобутку процесу для хмарних систем. Ці фази визначають процедуру на високому рівні абстракції. Цю структуру можна використовувати для стандартизації наскрізного процесу видобутку. Можна створювати інструменти для виконання спеціалізованих завдань, генеруючи модульні рішення.

Розглянемо три фази інтелектуального аналізу процесу. Ці фази забезпечують структуру абстракції високого рівня, яка визначає наскрізний механізм. Фазу вилучення, фазу трансформації та фазу аналізу можна побачити на рисунку 2.2.

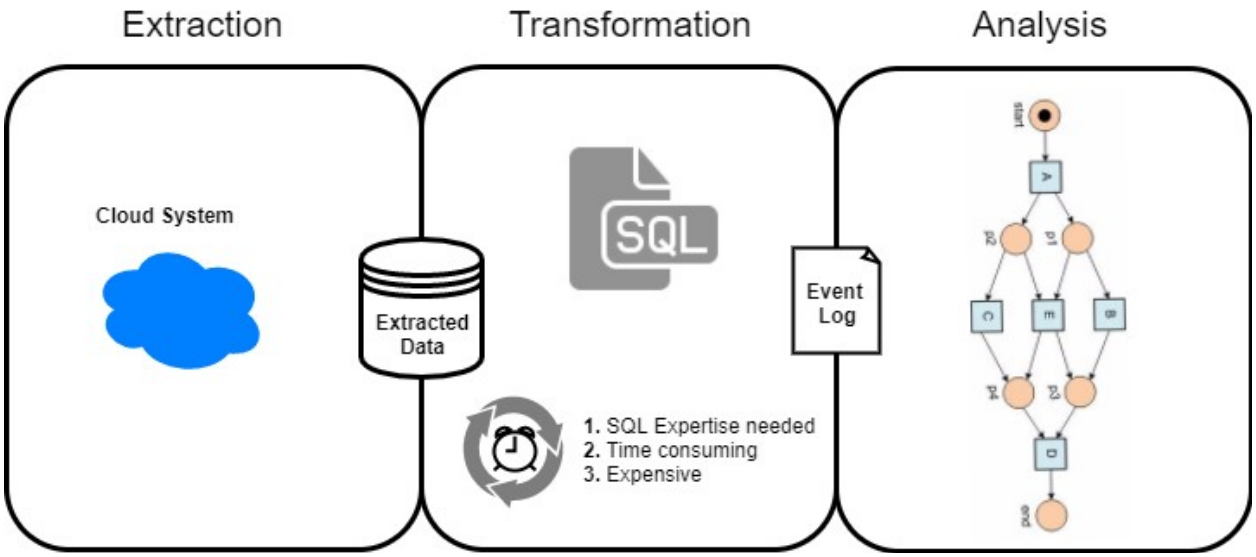


Рис. 2.2. Фази процесу майнінг у

2.3.1. Фаза вилучення (екстракції)

Фаза вилучення — це процес отримання інформації з хмарної системи. Метою цього кроку є збереження інформації в базі даних, до якої можна отримати доступ без подальшої взаємодії з хмарною системою. Це дуже залежить від цільової системи, оскільки кожен постачальник відповідає за

створення власних API. Автентифікація користувача, вибір даних і передача даних є ключовими компонентами цього етапу.

Існування API для взаємодії з хмарними системами відкриває двері для багатьох рішень. Ці інтерфейси забезпечують гнучкість, оскільки даними можна маніпулювати відповідно до конкретних цілей. Є два підходи до проектування екстракторів API. Перший підхід стосується вилучення всіх даних без урахування будь-якого конкретного вихідного формату. У цьому випадку добре підійде стиль NoSQL. Другий підхід полягає в додаванні обробки даних і маніпуляції з метою формування інформації в конкретній моделі даних. Другий підхід може використовувати реляційну базу даних або графову базу даних як вихідний формат.

У нашому випадку обраний підхід вилучення використовує другий варіант і реляційну базу даних як результат фази вилучення. Завдяки вилученню інформації у форматі реляційної бази даних на етапі трансформації передається менше накладних витрат. Крім того, традиційний підхід, представлений у першому розділі вимагає, щоб інформація зберігалася в реляційних базах даних. Отже, ми можемо визначити вхід і вихід цієї фази наступним чином:

- Введення: облікові дані автентифікації та виклики API.
- Вихід: необроблені дані у форматі реляційної бази даних.

2.3.2. Фаза трансформації

Фаза трансформації складається з набору операцій, необхідних для створення журналу подій із вилучених даних. Метою цієї роботи є порівняння двох методологій трансформації. Буде порівняно традиційне перетворення на основі сценаріїв SQL і автоматизоване перетворення на основі артефактоцентричного підходу. Обидва методи знаходяться в цій фазі. У Розділі 8 представлені отримані результати. Вхідні та вихідні елементи цієї фази можна визначити наступним чином:

- Вхідні дані: отримана інформація реляційної бази даних на етапі вилучення. Незалежно від того, який підхід використовується, вхідні дані однакові.

- Вихід: журнал подій. Ця фаза завершується, коли буде отримано журнал подій і можна розпочати аналіз аналізу процесів.

2.3.3. Фаза аналізу

На етапі аналізу до отриманого журналу подій застосовуються різні методи інтелектуального аналізу процесу. Метою цього етапу є отримання розуміння та значущої інформації, яку неможливо було отримати без аналізу процесів. Результати, отримані на цьому етапі, демонструють реальну цінність процесу інтелектуального аналізу. Фазу вилучення та фазу трансформації можна визначити як засоби, що дозволяють зробити фазу аналізу можливою. Вхід і вихід цієї фази можна визначити наступним чином:

- Вхід: журнал подій.

- Результат: Результатом цього етапу є різні висновки та ідеї.

2.4. Методика та інструменти для реалізації

У цьому розділі описано етап створення технічної реалізації, яка дозволяє виконувати вимоги, представлені в розділі 2.1. Пропоновану методику для аналізу даних показано на рисунку 2.3. Використання артефакто-орієнтованого підходу було особистим вибором. Мета цього розділу — пояснити, чому це налаштування відповідає початковим вимогам, а також представити кожен компонент та їхню взаємодію.

На рисунку 2.3 зображено три фази процесу інтелектуального аналізу процесів (Process Mining) з використанням артефактно-орієнтованого підходу:

1. Фаза вилучення (Extraction Phase):

(1) Джерела даних: Дані вилучаються з різних джерел, таких як Jira та Salesforce.

(2) API екстрактор: Спеціальний інструмент (API екстрактор) збирає дані з цих джерел.

2. Фаза трансформації (Transformation Phase):

(3) Артефактно-орієнтований підхід: Використовується для структурування та організації даних.

- База даних реплікації: Зберігає копії вихідних даних.

- База даних операцій з артефактами: Зберігає інформацію про зміни артефактів (об'єктів, пов'язаних з процесом).

- База даних зіставлення журналу подій: Відображає зв'язки між артефактами та подіями.

(4) SQL: Може використовуватися для запитів та маніпулювання даними на цьому етапі.

Взаємодії:

- Взаємодія 1: Дані з джерел завантажуються в базу даних вилучення.

- Взаємодія 2: Дані з бази даних вилучення використовуються для створення артефактів.

- Взаємодія 3: Інформація про артефакти використовується для створення журналу подій.

- Взаємодія 4: SQL може використовуватися для запитів та трансформації даних.

3. Фаза аналізу (Analysis Phase):

(5) Аналіз процесів: На цьому етапі використовуються різні методи інтелектуального аналізу процесів (Process Mining) для аналізу журналу подій та отримання знань про процес. Наприклад, виявлення вузьких місць, відхилень, оптимізація тощо.

Результат: Візуалізація процесу та аналітичні висновки (наприклад, діаграма процесу).

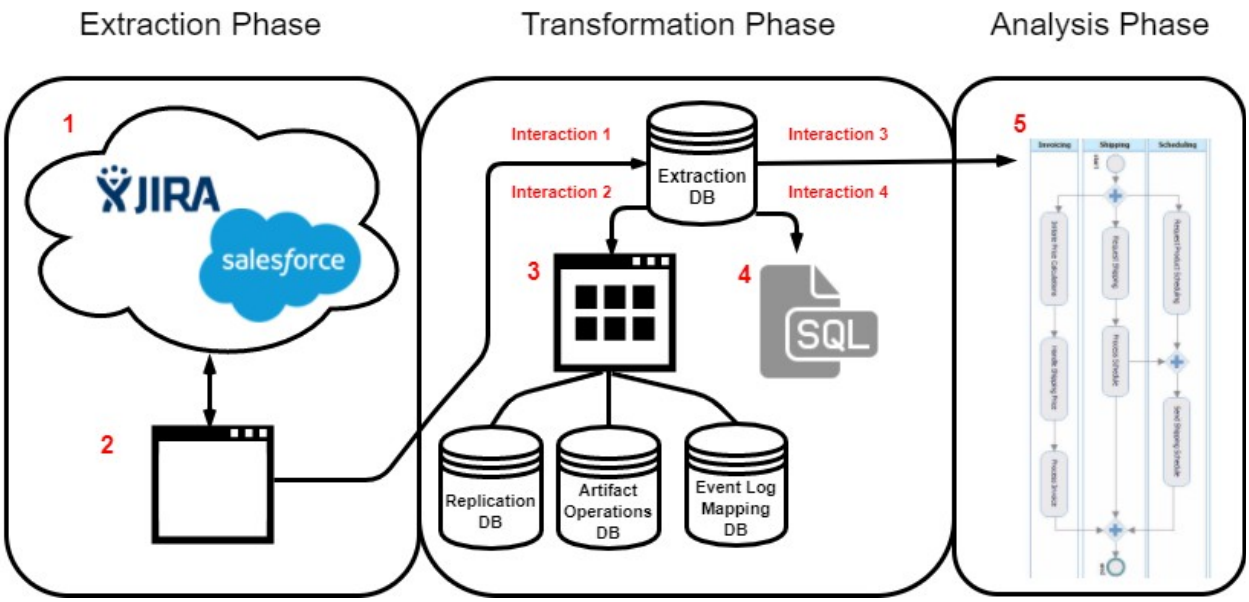


Рис. 2.3. Фази методики обробки даних

2.4.1. Хмарні системи

Для перевірки отриманих результатів у кількох системах використовуються дві хмарні системи. Перший приклад заснований на Jira. Jira — хмарна система, створена корпорацією Atlassian.

Таблиця 2.2.

Сімейство програм Jira

Name	Description
JIRA Software	JIRA Software is a software development planning tool that provides agile boards, development tool integrations, and reporting for teams using Scrum or Kanban.
JIRA Service Desk	JIRA Service Desk is a collaborative IT service desk with a powerful ticketing system, a self-service knowledge base and real-time reporting.
JIRA Core	JIRA Core is a streamlined issue tracker for business teams. Currently, JIRA Core does not have any application-specific functionality that you need to be aware of. JIRA Core's APIs, plugin modules, and project types are the same as those provided by the JIRA platform. There is no additional documentation for JIRA Core.

У таблиці 2.2 показано різні програми Jira. Jira Software стане цільовою системою для першої практичної демонстрації. Другий кейс – Salesforce. Salesforce зберігає інформацію для кількох процесів. У нашому випадку вибрано процес «Управління справами». Тому тільки було вилучено

пов’язані об’єкти та інформацію для управління справами. Salesforce for Service і Salesforce for Sales підтримують цей процес. API Extractor може працювати в обох системах, оскільки виклики API, об’єкти та дизайн однакові.

Таблиця 2.3.

Сімейство програм Salesforce

Name	Support Case Management Process
Sales Cloud	Yes
Service Cloud	Yes
Marketing Cloud	No
Community Cloud	No
Wave Analytics	No
App Cloud	No
IoT Cloud	No
Commerce Cloud	No

Для кожної хмарної системи необхідно розробити спеціальний екстрактор API. API Extractor, використаний для першого прикладу, був розроблений і реалізований за допомогою Java. Цей компонент відповідає REQ1.

2.4.2. Скрипти SQL

Сценарії SQL є традиційним механізмом для створення журналів подій. Подібно до API Extractor, при використанні традиційного підходу трансформації для кожної системи потрібно розробити окремий сценарій SQL. Сценарії SQL відповідають вимогам REQ2, REQ3, REQ10, але не відповідають вимогам REQ5, REQ6 і REQ7. Отже, слід розробити новий компонент, який також задовольняє REQ5 і REQ6.

Для вирішення незадоволених вимог REQ5 і REQ6 пропонується XTract. Цей програмний інструмент використовується для перевірки підходу, орієнтованого на артефакти, і відповідає за створення журналу подій на етапі трансформації. Оскільки існуючий інструмент має певні обмеження, буде

впроваджено нову версію. Цей інструмент відповідає вимогам REQ2, REQ3, REQ5, REQ6, REQ7, REQ10.

Буде надано аналіз аналізу процесів, щоб продемонструвати цінність аналізу процесів і переваги створення журналів подій.

2.4.3. Бази даних

На рисунку 2.4 можна побачити концептуальний дизайн бази даних вилучення та бази даних перетворення . Хоча можна використовувати будь-яку базу даних SQL, для тематичних досліджень MSSQL було обрано для бази даних вилучення, тоді як HyperSQL використовувався для бази даних трансформації. База даних трансформації створена за оригінальним дизайном XTract і не зазнає жодних змін.

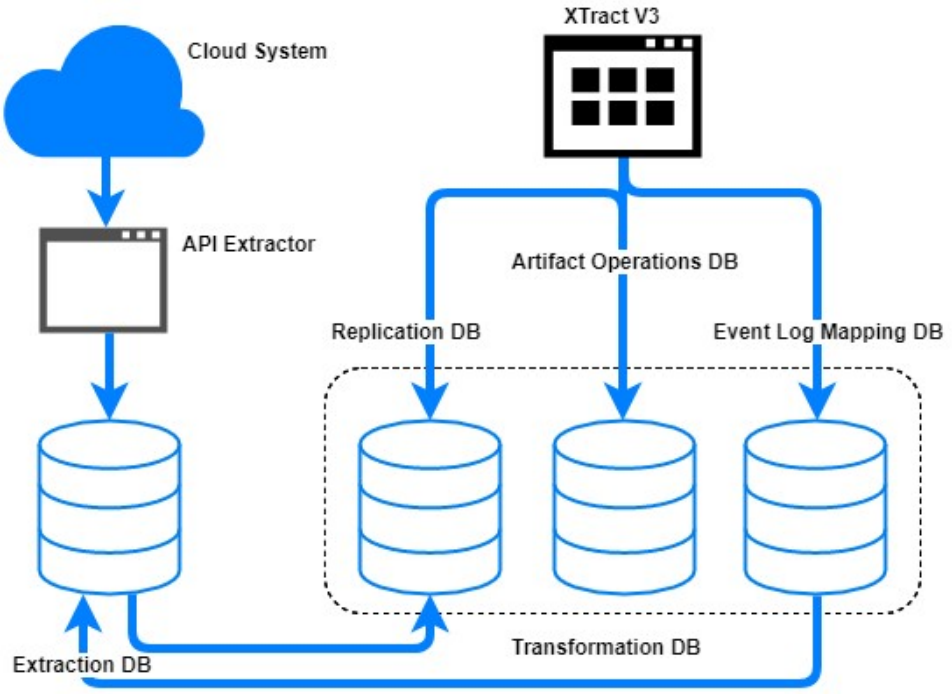


Рис. 2.4. База даних вилучення та база даних трансформації

База даних вилучення

На рисунку 2.3 взаємодіють три компоненти. A) API Extractor, B) SQL Scripts (традиційний підхід), C) XTract V3 (підхід, орієнтований на артефакти).

Взаємодія 1: База даних вилучення зберігає дані, отримані від API Extractor. Дизайн реляційної бази даних створюється в цьому компоненті та залежить від структури хмарної системи.

Взаємодія 2: для підходу, орієнтованого на артефакти, XTract підключиться до цієї бази даних і відтворить витягнуту інформацію в базу даних трансформації. Після створення XTract таблиці дій, яку він з'єднає, і перенесе її до бази даних вилучення.

Взаємодія 3: зв'язок між перетвореними даними та інструментом аналізу процесу.

Взаємодія 4: для традиційного підходу сценарії SQL буде розгорнуто в цій базі даних. Таблиця активності, створена за допомогою сценарію, буде зберігатися в цій базі даних.

База даних трансформації

Технічна схема налагодження (рис 2.3) ілюструє, як XTract взаємодіє з трьома різними базами даних. Комбінація цих баз даних відома як база даних трансформації, оскільки вони використовуються лише на етапі трансформації. Вони зберігають ту саму функціональність і дизайн, описані в [34]. Під час цієї роботи пропонується інша структура іменування (база даних реплікації, база даних операцій артефактів і база даних відображення журналу подій).

База даних реплікації

База даних реплікації – це копія, згенерована XTract із бази даних вилучення. Мета цієї реплікації — уникнути пошкодження даних.

База даних артефактів відповідає за зберігання метаданих бази даних реплікації, метрик для виявлення артефактів, посилань на таблиці, створених артефактів та їх структури.

База даних зіставлення журналу подій (база даних кешування) відповідає за зберігання згенерованих журналів подій, трас, дій і окремого

набору атрибутів. Він містить ту саму інформацію, що й файл XES, але у форматі реляційної бази даних.

2.5. Представлення та опис етапів модифікації як категорій інтелектуального аналізу процесу

Метою цього розділу є представлення різних етапів, на яких можуть бути реалізовані нові функції. Щоб відповідати всім вимогам і мати можливість трансформувати дані в журнал подій, було реалізовано необхідні функції. Як можна побачити в технічних налаштуваннях (рис. 2.3) , існує багато компонентів, які надають багато можливостей реалізації. З цієї причини необхідно визначити структуру, яка підсумовує різні можливості впровадження.

Етапи модифікації — це підкатегорії фаз інтелектуального аналізу процесу, які визначають, де слід розробляти технічні реалізації для перетворення даних у ланцюжку інструментів інтелектуального аналізу процесу. Він базується на фазах процесу інтелектуального аналізу, технічній настройці, оригінальному дизайні XTract, залучених базах даних і міркуваннях продуктивності.

Пропоновані етапи:

A) Етап вилучення,

B) Етап бази даних,

C) Етап артефакту,

D) Етап відображення журналу подій,

E) Етап відображення журналу подій,

F) Етап інструменту інтелектуального аналізу процесу.

У таблиці 2.4 відображається підсумок етапів модифікації та фази інтелектуального аналізу процесу, частиною якої вони є. У стовпці «Джерело даних» відображається компонент, який містить дані на певному етапі модифікації.

Таблиця 2.4.

Етапи модифікації, джерела даних і фази аналізу процесу

Modification Stage	Data Source	Process Mining Phase
Extraction Stage	Extraction Database	Extraction Phase
Replication Database Stage	Replication Database	Transformation Phase
Artifact Stage	Artifact Operations Database	Transformation Phase
Event Log Mapping Stage	Event Log Mapping Database	Transformation Phase
Event Log XES Stage	XES File	Transformation Phase
Process Mining Tool Stage	Event Log and Data Model	Analysis Phase

2.5.1. Етап вилучення даних

Модифікації етапу вилучення пов'язані з будь-якою операцією, застосованою до необроблених даних після їх отримання з хмарної системи та перед вставленням у базу даних вилучення. Деякі приклади таких операцій:

1. Ігнорування непотрібних об'єктів чи атрибутів.

2. Додавання, зміна або видалення типів даних з атрибута.

3. Додавання додаткової інформації, яка не включена в хмарну систему.

2.5.2. Етап реплікації бази даних

Зміни стадії бази даних реплікації пов'язані з будь-якою операцією, застосованою до даних після того, як вони потраплять у базу даних реплікації. Деякі приклади цих операцій (операції CRUD):

1. Видаліть стовпець.

2. Зміна назви таблиці.

3. Створення нових таблиць.

2.5.3. Етап артефакту

Модифікації стадії артефактів стосуються будь-якої операції, застосованої до виявленої структури артефактів. Деякі приклади таких операцій:

1. Додайте таблицю до артефакту.

2. Видалити дію з артефакту.

3. Змінити основну таблицю артефакту.

2.5.4. Етап відображення журналу подій

Зміни етапу відображення журналу подій пов’язані з будь-якою операцією, застосованою до бази даних відображення журналу подій. Деякі приклади таких операцій:

1. Видалити кейс.

2. Видалити атрибут із регістру.

3. Додайте дію до справи.

2.5.5. Журнал подій

Модифікації етапу XES журналу подій стосуються будь-якої операції, застосованої до файлу XES. Можливі зміни такі ж, як і на попередньому етапі. Різниця полягає в тому, що в цьому випадку зміни відбуваються у файлі XES.

1. Видалити атрибут із регістру.

2. Додайте дію до справи.

3. Етап інструменту процесу видобутку

Зміни стадії інструменту процесу інтелектуального аналізу пов’язані з будь-якою операцією, застосованою до даних у програмному забезпеченні процесу інтелектуального аналізу.

2.6. Опис фази трансформації даних

Існує два способи обробки виявлення домену на етапі вилучення. Деякі API мають можливість розрізняти конкретні типи даних для кожного значення. Якщо ця можливість недоступна, необхідно реалізувати виявлення домену. У цьому сценарії значні накладні витрати будуть додані на етапі вилучення. Виявлення домену також можна виконати на етапі реплікації бази

даних на етапі трансформації. Якщо це так, накладні витрати будуть додані на етапі трансформації, але не на етапі вилучення. Останньою особливістю є ідентифікація схеми бази даних. Застосовується та сама логіка накладних витрат, але є ще один важливий фактор. Беручи до уваги [26] і тематичні дослідження, можна сказати, що автоматична ідентифікація схеми бази даних працює не у всіх випадках. Автоматичне виявлення первинних і зовнішніх ключів є відкритою проблемою [1]. У таблиці 2.5 є шістнадцять функцій, які можна розвинути на етапі трансформації.

Таблиця 2.5.

Необхідні функції на етапі трансформації

ID	Feature Name	Possible Process Mining Phase	
H	Internal replication		Transformation
B	Ignore non-needed information	Extraction	Transformation
D	Domain discovery	Extraction	Transformation
E	Database schema identification	Extraction	Transformation
I	Automatic artifact discovery		Transformation
J	Manual artifact discovery		Transformation
K	Activities discovery.		Transformation
M	Attributes discovery.		Transformation
N	Removal of non-needed attributes.		Transformation
O	Removal of non-needed activities.		Transformation
P	Removal of repeated activities.		Transformation
Q	Activity naming		Transformation
R	Addition of attributes		Transformation
S	Activities granularity discovery		Transformation
T	Activities grouping	Analysis	Transformation
U	Activities table generation		Transformation

Реплікація буде виконана шляхом дублювання інформації з бази даних вилучення до бази даних реплікації. Це має бути оброблено на етапі реплікації бази даних. Є два варіанти виявлення артефактів. З одного боку, (I) автоматичне виявлення артефактів — це процес виявлення одного чи кількох артефактів без втручання користувача. З іншого боку, (J) виявлення артефакту вручну — це можливість вручну вибрати головну таблицю та решту таблиць, які відповідають артефакту. Як зазначено у вимогах, наразі

ми хочемо створити один журнал подій, тому наша мета полягає в тому, щоб виявити один артефакт. Відсутність автоматичного виявлення артефакту робить підхід напівавтоматичним, але дозволяє отримати цільовий артефакт. Це надзвичайно важливо, оскільки кінцевою метою є створення того самого журналу подій, що й традиційним способом. Крім того, без включення ручного виявлення артефакту можливість вибрати правильний ідентифікатор випадку залежатиме від моделі даних. Ідеальне рішення має запропонувати обидві можливості.

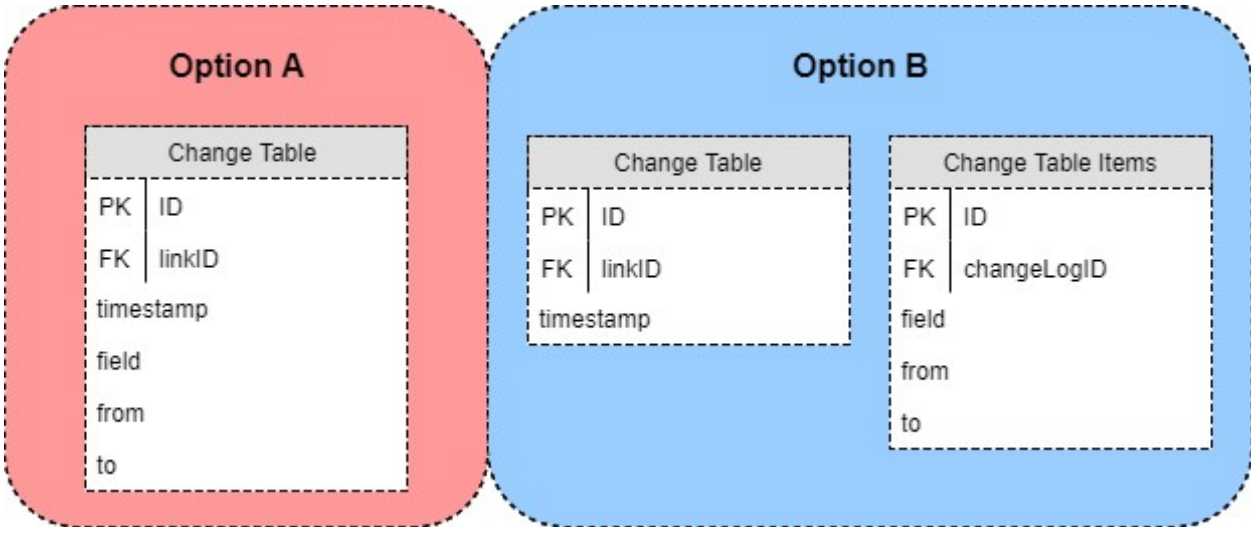


Рис. 2.5. Параметри представлення таблиці змін

У попередній роботі метою було виявити численні артефакти [26, 27], але було також реалізовано ручне створення артефактів. Обидві функції можна реалізувати лише на етапі артефакту.

У випадку діяльності важливо, щоб усі були виявлені. Атрибути можна додавати як додаткові ресурси в таблицю дій. Тому може знадобитися розширена функціональність для маніпулювання атрибутами. Виявлення дій і атрибутів можна виконувати лише на етапі артефакту.

Функції редагування потрібні, коли виявлено артефакт, його дії та атрибути. (N) Видалення непотрібних атрибутів і (O) непотрібних дій є ключовими для створення точного цільового журналу подій. Вони

дозволяють видалити інформацію, яка не повинна бути в цільовому журналі подій.

Також потрібне видалення повторюваних дій. Рисунок 2.6 відображення та приклад того, чому можуть повторюватися дії. Наприклад, значення стовпців CreatedDate або ModifiedDate також зберігаються в таблиці змін. Подібно до видалення непотрібних стовпців і таблиць, ці функції також покращують продуктивність і зменшують розмір сховища. Цю функціональність можна розробити на рівні артефактів, рівня відображення журналу подій або рівня журналу подій XES. В ідеалі їх слід видалити на рівні артефакту, перш ніж буде створено відображення журналу подій. Завдяки цьому видалення відбудеться до того, як будуть виявлені всі сліди, покращуючи продуктивність і зменшуючи розмір пам'яті.

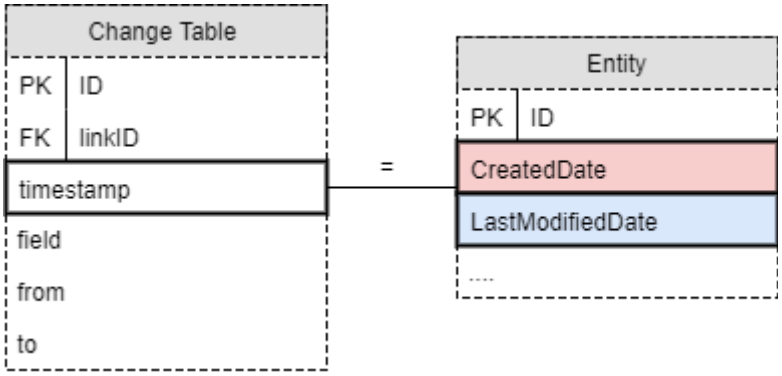


Рис. 2.6. Повторювані дії в хмарних системах

Назви діяльності — це можливість дати конкретну назву діяльності. Це додає рівень налаштування до журналів подій. Наприклад, журнали подій можна легко перекласти різними мовами. У попередній роботі ця функція була додана шляхом зміни інформації про необроблені дані, що означає на рівні вилучення або на рівні бази даних реплікації [26]. Наприклад, зміна назв стовпців у базі даних реплікації або додавання нових стовпців із назвами дій. Недоліком цього методу є те, що назви дій вводяться до виявлення дій. З цієї причини може бути важко визначити, що потрібно змінити. Інша можливість полягає в тому, щоб змінити назви дій на рівні артефакту, рівні

відображення журналу подій або рівні журналу подій XES. Ідеальне рішення полягає у зміні назв дій на рівні артефакту. Причина така ж, як і для (N) Видалення непотрібних атрибутів або (P) Видалення повторюваних дій. Статичні атрибути можна визначити як інформацію, яка пов'язана з діяльністю та ніколи не змінюється.

Для включення сортування необхідно додати статичні атрибути. Сортування потрібне у випадках, коли різні дії відбуваються одночасно (одне значення позначки часу). Тому це статичний атрибут, оскільки він завжди буде однаковим у різних видах діяльності. Статичні атрибути можуть бути включені на рівні бази даних, рівні артефакту, рівні відображення журналу подій або рівні журналу подій.

Виявлення деталізації дій — це можливість розгортати або згортати ту саму діяльність шляхом налаштування рівня абстракції. На рисунку 2.7 наведено наочний приклад розгорнутої дії «Зміна статусу». Ту саму дію можна розділити на «Змінити пріоритет на X» або «Змінити пріоритет з Y на X» . Необхідна інформація присутня на рівні артефакту, рівні відображення журналу подій і рівні журналу подій. Недоліки впровадження рішення на рівні зіставлення журналу подій або на рівні XES журналу подій полягають у тому, що дії вже виявлено та всі траси згенеровано. В ідеалі деталізацію дій можна контролювати, виявляючи їх на рівні артефакту.

CASE_ID	ACTIVITY_NAME	EVENT_TIME
5002400000FpgelAAB	Create Case	2016-02-12 11:36:57.000
5002400000FpgelAAB	Change Owner	2016-02-12 11:36:59.000
5002400000FpgelAAB	Change Status to Working	2016-02-12 13:46:24.000
5002400000FpgelAAB	Change Status to Awaiting Reply	2016-02-12 17:10:23.000
5002400000FpgelAAB	Change Status to On Hold	2016-02-15 15:55:37.000
5002400000FpgelAAB	Change Priority to Low	2016-02-16 11:43:49.000
5002400000FpgelAAB	Change Status to Awaiting Reply	2016-03-01 12:59:48.000
5002400000FpgelAAB	Change Status to On Hold	2016-03-05 10:32:51.000
5002400000FpgelAAB	Change Status to Awaiting Reply	2016-04-25 15:26:25.000
5002400000FpgelAAB	Change Status to Closed	2016-05-05 11:35:26.000

Рис. 2.7. Приклад деталізації дій

Групування дій є повною протилежністю виявлення деталізації дій. У цьому випадку мета полягає в тому, щоб об’єднати дві чи більше діяльності в одну. Знову ж таки, це можна розробити на рівні артефакту, на рівні відображення журналу подій або на рівні журналу подій. Дотримуючись тих самих міркувань, наведених для попередніх функцій, доцільніше керувати групуванням до створення журналу подій. Тому найкращим варіантом є вирішення цієї вимоги на рівні артефакту. Також можна розв’язати групування дій на рівні інструменту інтелектуального аналізу процесу.

Недоліком цього варіанту є те, що не всі інструменти інтелектуального аналізу процесів можуть групувати дії, тому це менш загальне рішення.

Останньою необхідною функцією на етапі трансформації є створення таблиці дій. Створення таблиці дій відповідає створенню згаданої таблиці та параметрам експорту. Окрім ідентифікатора справи, назви діяльності та позначки часу, слід додати інші атрибути. У цьому випадку функціональні можливості потрібно створити після створення журналу подій. Тому його можна додати на рівні журналу подій XES або на рівні відображення журналу подій. Недоліки його реалізації на рівні журналу подій XES полягають у тому, що потрібно реалізувати аналізатор файлів XES. Запит до бази даних зіставлення журналу подій ефективніший, ніж взаємодія з файлом XES.

Висновки до розділу

Другий розділ присвячено методам, методиці та вимогам до систем обробки й видобування даних із журналів подій у хмарних системах. Описано особливості процесу інтелектуального аналізу (Process Mining) для хмарних систем, що дає можливість виявлення та покращення бізнес-процесів на основі даних, зібраних у журналах подій. Це має особливу значущість у контексті хмарних технологій, які генерують значні обсяги даних. Визначено основні вимоги до побудови журналу подій для хмарних

систем, що включають точність, повноту, своєчасність збору даних, їх структурованість та можливість обробки в умовах масштабованих хмарних середовищ. Ці вимоги забезпечують ефективний та результативний процес аналізу.

Описано три основні фази майнінгу в хмарних системах: Фаза вилучення, яка відповідає за збір даних із різних джерел, Фаза трансформації, що полягає у підготовці та структуризації даних для подальшого аналізу, Фаза аналізу, яка спрямована на виявлення та інтерпретацію результатів. Представлено методику й інструменти, що забезпечують ефективне виконання зазначених фаз. До основних інструментів належать хмарні системи для зберігання та обробки даних, SQL-скрипти для маніпуляції даними, а також бази даних і спеціалізовані реплікаційні бази для надійного зберігання та резервування даних.

Описано категорії інтелектуального аналізу процесу, зокрема етапи вилучення даних, реплікації бази, формування артефактів та відображення подій у журналі. Ці етапи забезпечують послідовність і структурованість обробки даних. Деталізовано процес трансформації даних для подальшого аналізу, включаючи методи нормалізації, агрегування та відображення даних у форматі, придатному для інтелектуального аналізу.

Розділ надає чітке уявлення про послідовність етапів, методику та інструменти для побудови ефективних систем обробки журналів подій у хмарних системах, що є основою для точного аналізу та вдосконалення бізнес-процесів.

РОЗДІЛ 3. ІМПЛЕМЕНТАЦІЯ МЕТОДІВ ТА АЛГОРИТМІВ ВИДОБУВАННЯ ДАНИХ ДЛЯ ПОБУДОВИ ЖУРНАЛУ ПОДІЙ ХМАРНИХ СИСТЕМ

3.1. Представлення основних компонент

На рисунку 3.1 можна побачити різні компоненти екстрактора API та зовнішні сутності. Компоненти екстрактора API представлені чотирма блоками, а зовнішніми об’єктами є хмарна система та база даних реплікації.

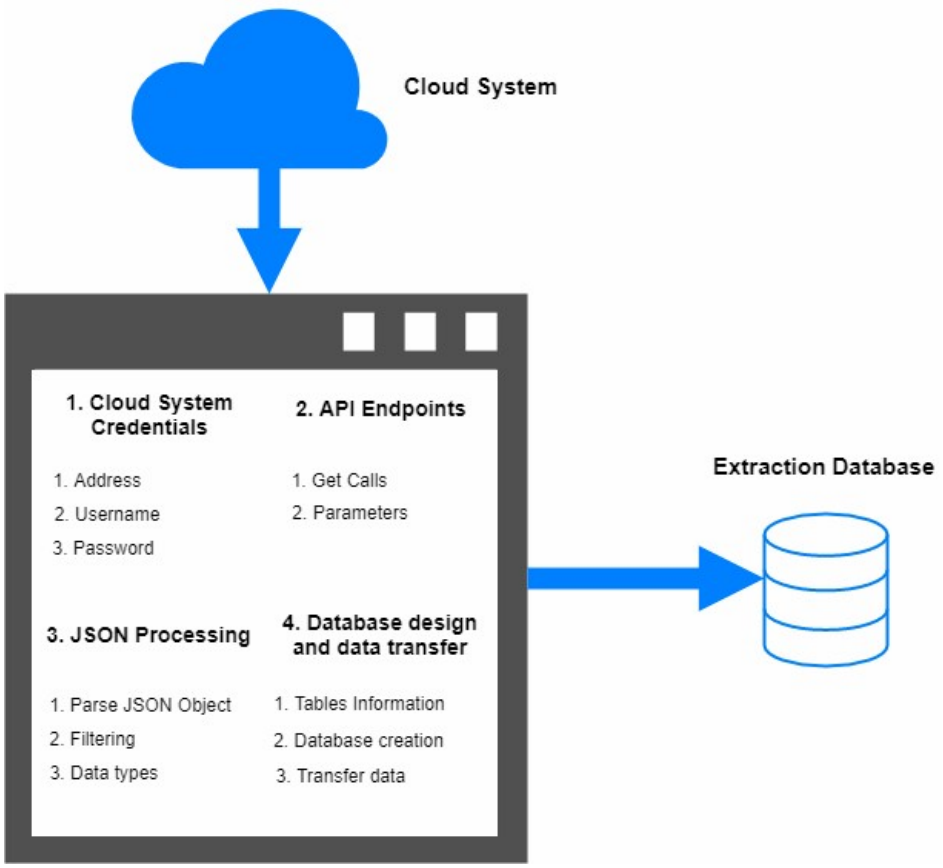


Рис. 3.1. Огляд екстрактора API

При розробці API Extractor необхідно враховувати хмарну систему та базу даних реплікації. З одного боку, хмарна система є фундаментальною, оскільки її впровадження впливає на загальний дизайн. Такі елементи, як механізм автентифікації, доступність інформації, дизайн реляційної бази

даних і базова технологія обміну даними, суворо залежать від інтерфейсу хмарної системи. З іншого боку, бажану технологію реплікації бази даних потрібно реалізувати в API Extraction, вибравши правильний драйвер бази даних. MSSQL буде використовуватися як база даних реплікації.

Різні частини API-екстрактора розділені на блоки з певними завданнями. Наступне пояснення базується на Jira Software API Extractor, оскільки його було створено з нуля. Тим не менш, відмінності між цим екстрактором і екстрактором Salesforce будуть згадані.

1. Облікові дані хмарної системи

Цей блок відповідає за аутентифікацію користувача. Як зазначалося раніше, автентифікація залежить від опцій, які надає хмарна система. Для обох екстракторів використовувалася базова автентифікація. Отже, потрібні лише ім’я користувача, пароль та адреса. Цей механізм автентифікації зменшує накладні витрати порівняно з іншими доступними варіантами.

2. Кінцеві точки API

Кінцеву точку API можна визначити як інтерфейс, який забезпечує передачу інформації між API Extractor і хмарною системою. В API Rest цей канал має форму URL-адреси, де доступні дані об’єкта. Крім того, можуть бути доступні різні параметри для формування бажаної реакції з метою зменшення або збільшення бажаної інформації. У цьому блоці програмуються та стають доступними потрібні виклики API. Для наших прикладів було створено лише бажані кінцеві точки, тоді як кінцеві точки, які повертали непотрібну інформацію, були виключені. Для першого прикладу бібліотека 3 використовувався для скорочення часу розробки. Ця бібліотека обробляла доступні кінцеві точки, параметри та відповіді. У випадку Jira Software API кінцеві точки створюють відповідь у форматі JSON.

У лістингу 3.1 показано анонімний приклад відповіді JSON після виконання виклику кінцевої точки API. Оскільки до виклику API не було включено параметрів, відповідь міститиме всі проекти, доступні в системі.

Лістинг 3.1. Відповідь API.


```
[{"expand":"description,lead,url,projectKeys",
"self":"https://jira/rest/api/2/project/10502",
"id":"10502",
"key":"XXX1",
"name":"Name1",
"avatarUrls":{"48x48":"https://jira/secure/projectavatar?avatarId=10903",
"24x24":"https://jira/secure/projectavatar?size=small&avatarId=10903",
"16x16":"https://jira/secure/projectavatar?size=xsmall&avatarId=10903",
"32x32":"https://jira/secure/projectavatar?size=medium&avatarId=10903"},
"projectTypeKey":"typeKey1"},
{"expand":"description,lead,url,projectKeys",
"self":"https://jira/rest/api/2/project/10301",
"id":"10301",
"key":"XXXXX",
"name":"Project Name 1",
"avatarUrls":{"48x48":"https://jira/secure/projectavatar?pid=10301&avatarId=10011",
"24x24":"https://jira/secure/projectavatar?size=small&pid=10301&avatarId=10011",
"16x16":"https://jira/secure/projectavatar?size=xsmall&pid=10301&avatarId=10011",
"32x32":"https://jira/secure/projectavatar?size=medium&pid=10301&avatarId=10011"},
"projectTypeKey":"software"},
{"expand":"description,lead,url,projectKeys",
"self":"https://jira/rest/api/2/project/10200","id":"10200",
"key":"XXX2",
"name":"Project Name 2",
"avatarUrls":{"48x48":"https://jira/secure/projectavatar?pid=10200&avatarId=10011",
"24x24":"https://jira/secure/projectavatar?size=small&pid=10200&avatarId=10011",
"16x16":"https://jira/secure/projectavatar?size=xsmall&pid=10200&avatarId=10011",
"32x32":"https://jira/secure/projectavatar?size=medium&pid=10200&avatarId=10011"},
"projectTypeKey":"typeKey2"}]
```


3. Обробка JSON

Згадана бібліотека містить усі доступні об’єкти Jira Software у класах Java, що забезпечує ефективний механізм перетворення корисного навантаження JSON на об’єкти Java. Після того, як інформація збережена в об’єктах Java, її можна зіставити з бажаною структурою реляційної бази даних. Фактичне зіставлення з базою даних відбувається в останньому блоці. Блок обробки JSON також відповідає за фільтрацію непотрібних атрибутів, які не були відфільтровані за допомогою викликів кінцевих точок. Нарешті, на цьому кроці можна обробити визначення типів даних. Для нашого випадку бібліотека містила необхідні типи даних, уже визначені в атрибутах об’єкта Java. У випадку Salesforce API надає доступ до конкретних типів даних об’єктів.

У лістингу 3.2 можна побачити приклад об’єкта. У цьому випадку об'єкт представляє проект. Вимога полягає в тому, щоб атрибути відповідали тому самому формату, що й у корисному навантаженні JSON. Цей формат

пов’язаний із типом структури, яка містить інформацію. Наприклад, використання Java Array, Java Map чи іншої структури має відповідати структурі корисного навантаження JSON. Крім того, можна визначити типи даних кожного елемента, які можна використовувати при створенні реляційної бази даних.

Лістинг 3.2. Приклад об’єкта Java

```
public class Project extends Resource {

    private Map<String, String> avatarUrls = null;
    private String key = null;
    private String name = null;
    private String description = null;
    private User lead = null;
    private String assigneeType = null;
    private List<Component> components = null;
    private List<IssueType> issueTypes = null;
    private List<Version> versions = null;
    private Map<String, String> roles = null;
    private ProjectCategory category = null;
    private String email = null;

    /**
     * Creates a project from a JSON payload.
     *
     * @param restclient REST client instance
     * @param json JSON payload
     */
    protected Project(RestClient restclient, JSONObject json) {
        super(restclient);

        if (json != null)
            deserialise(json);
    }
```

Останнім завданням, яке необхідно виконати в цьому блоці, є передача даних в базу даних реплікації. Для цього використовувався спеціальний конектор бази даних. За допомогою цього драйвера можна виконувати різні операції, наприклад визначення та створення таблиць, стовпців і типів даних. Крім того, у цьому блоці дані передаються до бази даних реплікації.

3.2. Представлення необхідних функцій на етапі вилучення

У цьому розділі буде обговорено представлене рішення для функцій фази вилучення та пов’язано з конкретним блоком, який вирішує функцію з рисунка 3.1.

A) Інформація, що зберігається у форматі реляційної бази даних

Для обох прикладів кожна система представляла інформацію, організовану в окремі об’єкти. Кожен об’єкт містить власний ідентифікатор і чітко пов’язаний з іншими об’єктами. Таким чином, витягнуті дані можуть бути перетворені в реляційну базу даних. Тим не менш, дизайн реляційної бази даних і відображення елементів потрібно обробляти правильно. Ця функція обробляється між блоками 3 і 4. У блоці обробки JSON необхідні значення зберігаються в об’єктах Java і відображаються в таблицях реляційної бази даних.

B) Ігнорування непотрібної інформації

Щоб ігнорувати непотрібну інформацію, слід використовувати лише необхідні виклики API. Залежно від рівня складності API можуть бути доступні кілька варіантів фільтрації відповідей кінцевих точок. Для нашого першого прикладу API має розширені можливості пошуку. За допомогою цієї опції можна створювати дуже специфічні виклики API. Це рішення входить до блоку 2.

Якщо розширені можливості пошуку відсутні, інший підхід полягає в тому, щоб ігнорувати непотрібне формування під час відображення атрибутів у реляційну базу даних. При проектуванні реляційної бази даних слід враховувати лише релевантну інформацію. Це рішення включено в блок 4.

C) Зміна представлення таблиці

Як було зазначено, таблиці змін зберігають усі зміни, які відбуваються в системі. З цієї причини ці таблиці є основним джерелом для отримання активності. Отже, підтримку обох варіантів було додано до екстрактора API. Дизайн та відображення таблиць змін обробляються між блоками 3 і 4.

D) Виявлення домену

Виявлення домену можна успішно обробляти, визначаючи кожен окремий об’єкт Java.

E) Ідентифікація схеми бази даних

Незважаючи на те, що схему бази даних можна було додати на рівні вилучення, було прийнято рішення використовувати цю функцію в XTract. Основна причина полягає в тому, щоб уникнути накладних витрат у API Extractor. Крім того, наявність можливості ідентифікувати схему бази даних на етапі трансформації є більш загальним рішенням. В іншому випадку кожен API Extractor повинен містити цю функцію.

F) Автентифікація API

Вибраний метод автентифікації – базова автентифікація HTTP на основі файлів cookie та OAuth. Недоліком цього варіанту автентифікації є те, що він менш безпечний. У будь-якому випадку безпека не є головною проблемою для поточного проекту.

G) Передача даних

Створено можливість передачі даних у певну базу даних шляхом надання правильних облікових даних. Для першого прикладу дані передаються до бази даних вилучення за допомогою драйверів JDBC. Базу даних MSSQL було вибрано, оскільки вона підтримується XTract.

3.3. Реалізація отримання даних з журналу подій на основі артефакто-орієнтованого підходу

В цьому розділі ми обговорюємо, як ми розширили підхід, орієнтований на артефакти, і XTract V1, щоб реалізувати відсутні функції, в існуючих системах. Буде згадано конкретні оригінальні кроки орієнтованого на артефакти підходу, у які було внесено зміни.

Замість того, щоб розглядати процес як послідовність дій (як у традиційному підході), артефактно-орієнтований підхід зосереджується на тому, як змінюється стан артефактів під час процесу. Кожен артефакт має життєвий цикл, і процес можна представити як сукупність змін станів різних артефактів. Етап модифікації, на якому розробляється доповнення, як

описано в другому розділі також буде згадано. На рисунку 3.2 представлено відображає оригінальний артефакто-орієнтований підхід.

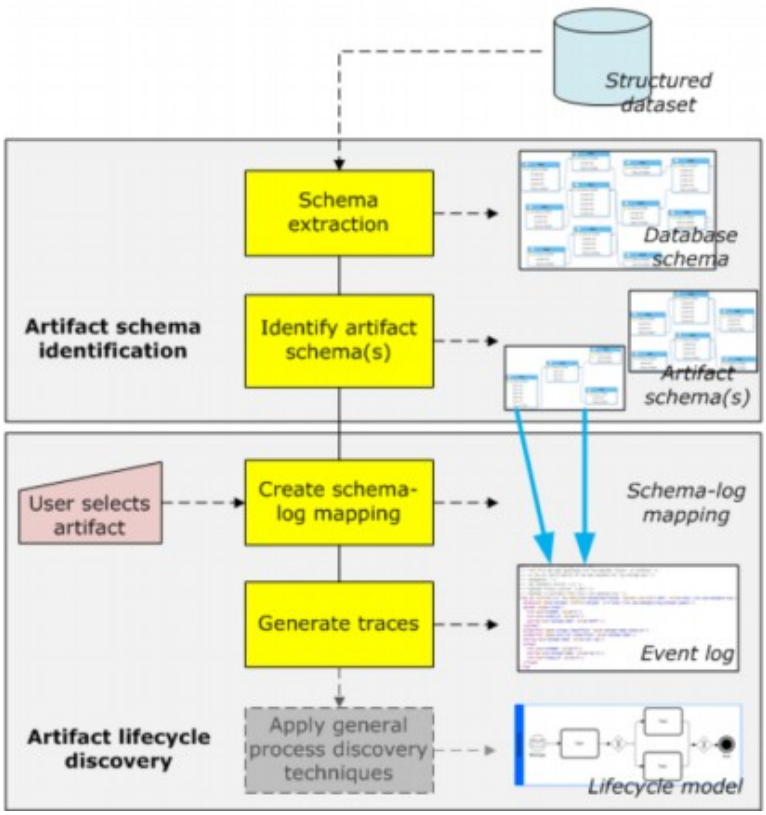


Рис. 3.2. Класичний метод артефакто-орієнтованого підходу

3.3.1. Підготовка даних

Метою цього кроку є додавання можливостей редагування витягнутої інформації. За допомогою правильного набору операцій інформацію можна трансформувати таким чином, щоб артефакто-орієнтованИЙ підхід міг створити цільовий журнал подій. Крок маніпулювання даними вирішує наступні необхідні функції: (B) ігнорування непотрібної інформації, (K,S) виявлення дій і виявлення деталізації дій. Усі ці функції були реалізовані на етапі реплікації бази даних.

Екстрактор може видаляти непотрібні таблиці та стовпці, тому реалізує функцію непотрібної інформації. Ця функція покращує продуктивність і зменшує розмір пам’яті. Екстрактор надає графічний інтерфейс, за допомогою якого користувачі можуть вибирати таблиці та стовпці, які слід

видалити. Після вибору таблиці або стовпця генерація запиту буде виконана автоматично. На рисунку 3.3 показані інтерфейси користувача.

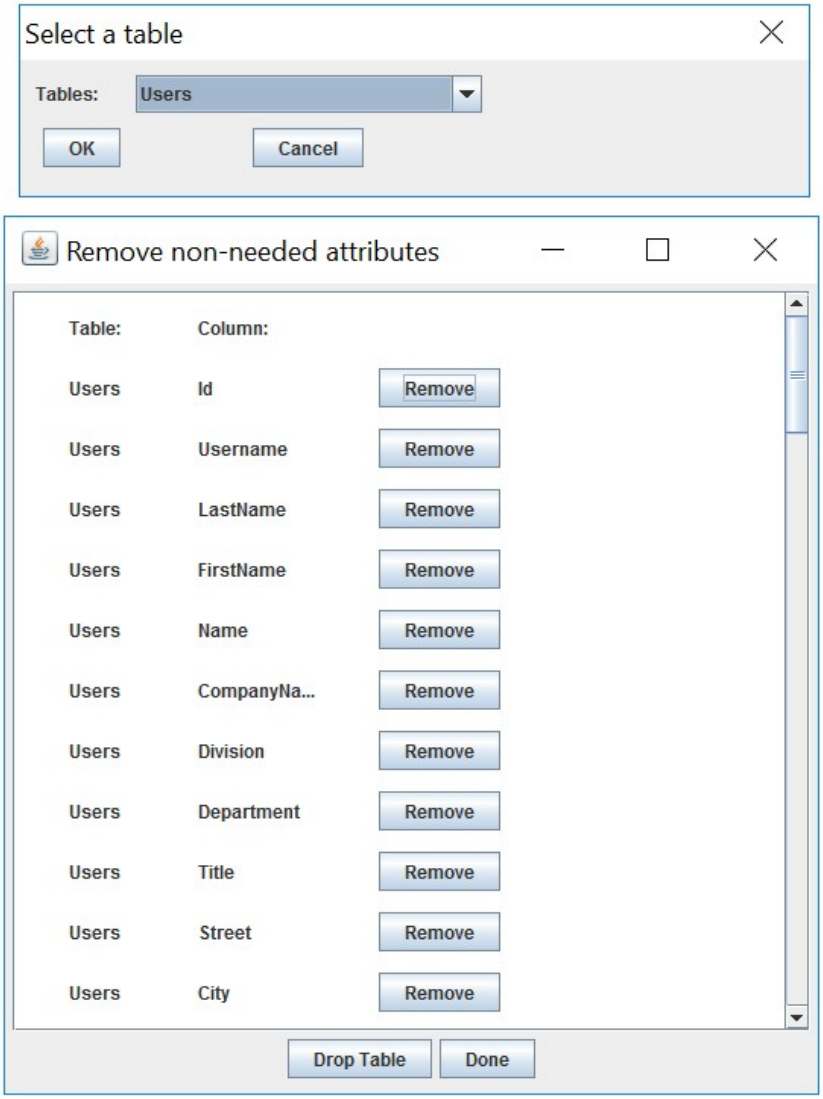


Рис. 3.3. Інтерфейс видалення таблиць або полів

Як зазначалося раніше, основною проблемою при виявленні активностей в екстракторі є ефект вертикального антирозбиття. На рисунку 3.4 показано приклад таблиці, що містить кілька сутностей. "Розділення" таблиць на окремі таблиці для кожної сутності вирішить проблему виявлення активностей. У розділі 2 (рисунок 2.5) можна знайти варіанти представлення таблиць змін. В обох випадках стовпець поля може бути використаний для розділення таблиці на кілька таблиць.

Id	IsDeleted	CaseId	CreatedById	CreatedDate	Field	OldValue	NewValue
01724...	0	500240000...	0052400000...	2016-02-15 10:...	Contact		
01724...	0	500240000...	0052400000...	2016-09-26 16:...	Contact		
01724...	0	500240000...	0052400000...	2016-06-06 14:...	Contact		
01724...	1	500240000...	0052400000...	2017-04-06 11:...	Contact		
01724...	1	500240000...	0052400000...	2017-04-06 11:...	Contact		
01724...	0	500240000...	0052400000...	2016-02-02 17:...	CurrencyIsoCode		
01724...	0	500240000...	0052400000...	2016-06-17 18:...	Description		
01724...	0	500240000...	0052400000...	2016-06-17 18:...	Description		
01724...	0	500240000...	0052400000...	2016-06-01 18:...	Description		
01724...	0	500240000...	0052400000...	2016-06-01 18:...	Description		

Рис. 3.4. Кілька об’єктів, представлених в одній таблиці

Визначення 3.1. Стовпець поля можна визначити як стовпець, що містить об'єкт або сутність, яка змінилася. Назва стовпця може відрізнятися в різних системах.

Під час досліджень було виявлено, що можливі системи з кількома таблицями змін (тобто чотири таблиці змін, представлені Варіантом 1, або дві таблиці змін, представлені Варіантом 2). Таким чином, екстрактор може розділити кілька таблиць змін, представлених або Варіантом 1, або Варіантом 2. У випадку Варіанта представлення 2, перед цим необхідно виконати операцію з'єднання.

Лістинг 3.3 показує алгоритм з'єднання цих таблиць. Вхідні дані має надати користувач. Алгоритм вирішення вертикального антирозбиття представлений у лістингу 3.4. Цей алгоритм також містить визначення рівнів деталізації, які обговорюються далі.

Лістинг 3.3. Об’єднання псевдокоду таблиць змін (Варіант 2).

```
def joinChangesTables():

  var changesTable1    #First Table
  var changesTable2    #Second Table containing "From" and "To" Columns
  var key1             #Reference key from changeTable1 connecting both tables
  var key2             #Reference key from changeTable2 connecting both tables

  var mergedTable      #Result of joining both tables

  mergedTable = JOIN(changeTable1 AND ChangeTable2 ON Key1=Key2)

  return mergedTable
```

Лістинг 3.4. Розділені таблиці та псевдокод визначення рівнів деталізації

```
def splitTable():

  var tableToSplit           #Table with Vertical Anti-Partitioning
  var fieldColumn            #Field column
  var fromColumn             #Column holding previous value
  var toColumn               #Column holding new value

  var highGranularityFields  #Store user selections
  var lowGranularityFields   #Store user selections
  var noGranularityFields    #Store user selections

  var newTablesList          #New table results

  for value in fieldColumn:
   if(value is selected by user as High Granularity)
     highGranularityFields.add(value)

   if(value is selected by user as Low Granularity)
     lowGranularityFields.add(value)

   if(value is selected by user as No Granularity)
     noGranularityFields.add(value)

  for field in highGranularityFields:
     newTable.setName(S_field_FROM_fromColumn_TO_toColumn)
     newTable =(INSERT ALL WHERE tableToSplit.field=field FROM tableToSplit)
     newTablesList.add(newTable)

  for field in lowGranularityFields:
     newTable.setName(S_field_TO_toColumn)
     newTable =(INSERT ALL WHERE tableToSplit.field=field FROM tableToSplit)
     newTablesList.add(newTable)

  for field in noGranularityFields:
     newTable.setName(S_field)
     newTable =(INSERT ALL WHERE tableToSplit.field=field FROM tableToSplit)
     newTablesList.add(newTable)

  return newTablesList
```

Перевагою таблиць змін є те, що вони зберігають дуже детальну інформацію про те, що змінилося. Стовпці «від» і «до» можна використовувати для визначення попереднього та нового значення. Ця інформація потрібна для встановлення рівня деталізації.

Визначення 3.2. (Стовпець From) Стовпець From можна визначити як стовпець, який містить значення до зміни.

Визначення 3.3. (до стовпця) До стовпця можна визначити як стовпець, який містить значення після зміни.

Таблиця 3.1.

Пропоновані рівні деталізації

Granularity Level Name	Columns Used	Example
No Granularity	None	Change Status
Low Granularity	To	Change Status to Done
High Granularity	From	Change Status from Open to Done

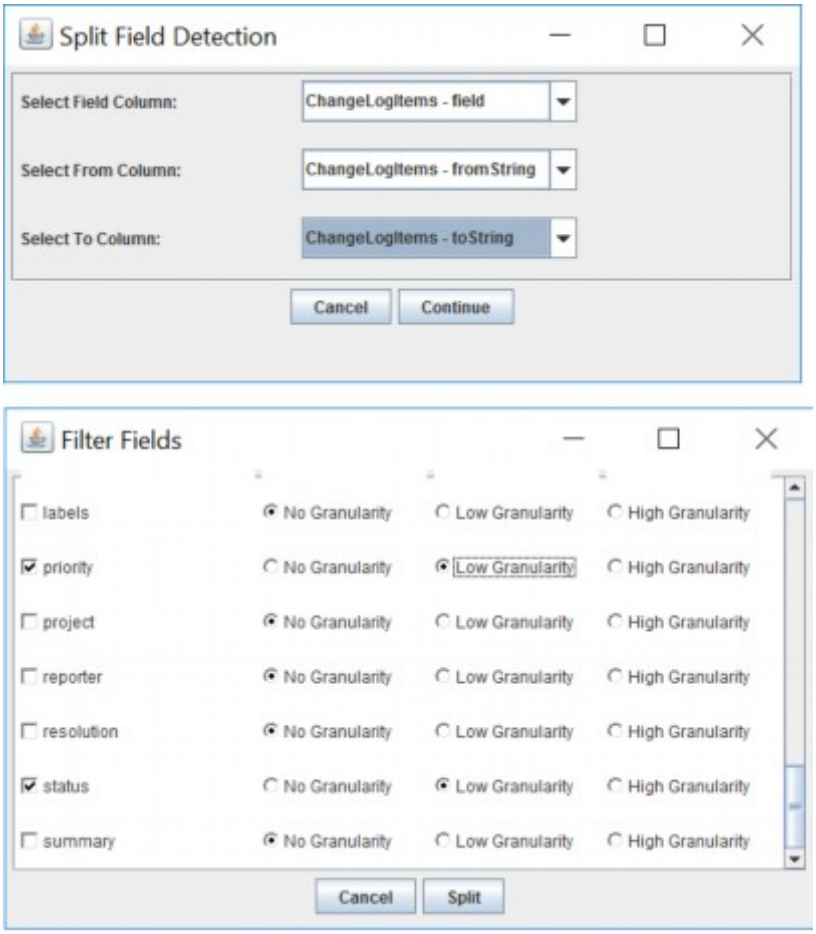


Рис. 3.4. Вибір гранулярності

Використовуючи ці стовпці, можна створити три рівні деталізації для найменування активностей. Запропоновані рівні можна побачити в таблиці 3.1. Поєднуючи цю ідею з функціональністю розділення за полем, можна створити функцію виявлення деталізації активностей.

Для визначення рівнів деталізації активностей екстрактор надає графічний інтерфейс, де потрібно визначити стовпці "з", "до" та "поле". За допомогою цієї інформації він створить всі поля, які були змінені. Потім

користувач може вибрати поля, які його цікавлять, і рівень деталізації для кожного поля. Залежно від цього вибору будуть створені таблиці для всіх можливих варіантів.

На рисунку 3.4 показано приклад розділення таблиць з рівнем деталізації. Назви цих таблиць і стовпців можуть відрізнятися в різних системах. У наведеному прикладі таблиця змін розділена на кілька таблиць, що містять всю інформацію лише про пріоритети та статуси. Оскільки деталізація була визначена як "Низька" для обох полів, згенеровані таблиці будуть названі відповідно до їх значень. У таблиці 3.3 показано можливий результат розділення. Екстрактор додає "S" перед таблицями, які були створені в результаті операції розділення. Після цієї операції розділення активності з таблиць змін зберігаються у відповідний спосіб для виявлення на наступних кроках.

Таблиця 3.2.

Приклад назв після розбиття

New table name
S_status_TO_Todo
S_status_TO_Open
S_status_TO_Released
S_priority_TO_High
S_priority_TO_Critical
S_priority_TO_Low

3.3.2. Ідентифікація схеми бази даних

Потрібна ручна ідентифікація схеми бази даних, оскільки автоматична ідентифікація схеми бази даних не дає очікуваних результатів. Це було помічено в [26], але також підтверджено під час роботи над тематичними дослідженнями. Відповідно, у екстракторі реалізовано ручну ідентифікацію первинних і зовнішніх ключів. На рисунках 3.5 і 3.6 показані ручні первинні ключі та ідентифікація зовнішніх ключів відповідно. Обидва механізми були реалізовані на етапі реплікації бази даних.

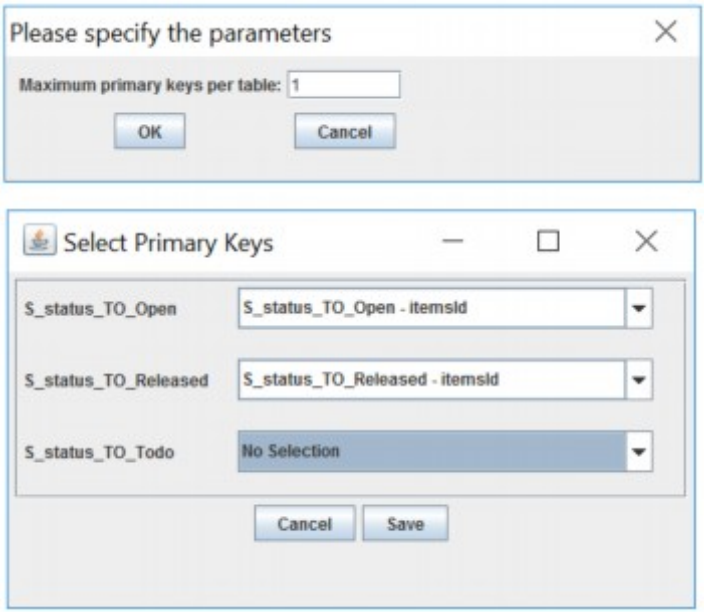


Рис. 3.5. Вибір первинних ключів

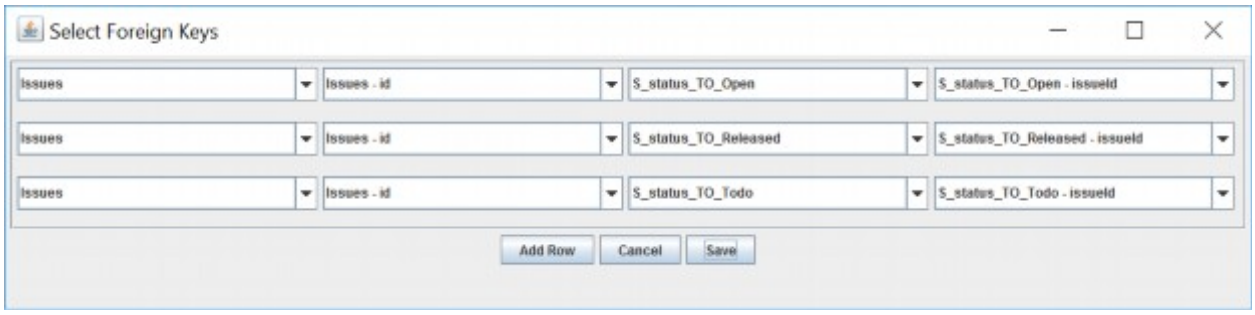


Рис. 3.6. Вибір вторинних ключів

Можна одночасно використовувати автоматичне та ручне виявлення первинного та зовнішнього ключів. Наприклад, використовуючи методи автоматичного виявлення, вже реалізовані, і доповнення відсутності ключів вручну. Обидва механізми зберігають первинні та зовнішні ключі в базі даних операцій артефактів.

Метою кроку маніпуляції артефактами є надання можливостей редагування виявлених артефактів, їх діяльності та атрибутів. Його слід включити як частину етапу ідентифікації схеми артефактів у підході, орієнтованому на артефакти. Запропоновані кроки вирішують наступні необхідні функції:

- Ручне виявлення артефакту,

- видалення непотрібних атрибутів,

- видалення непотрібних дій,

- видалення повторюваних дій,

- діяльність іменування,

- додавання статичних атрибутів,

- групування дій.

Усі ці функції були реалізовані на стадії артефактів.

Крок створення таблиці операцій експорту є повним досягненням двох цілей. Першою метою є створення таблиці діяльності. Таблиця дій складається з ідентифікатора справи, назви дії та позначки часу, а також містить додаткові атрибути.

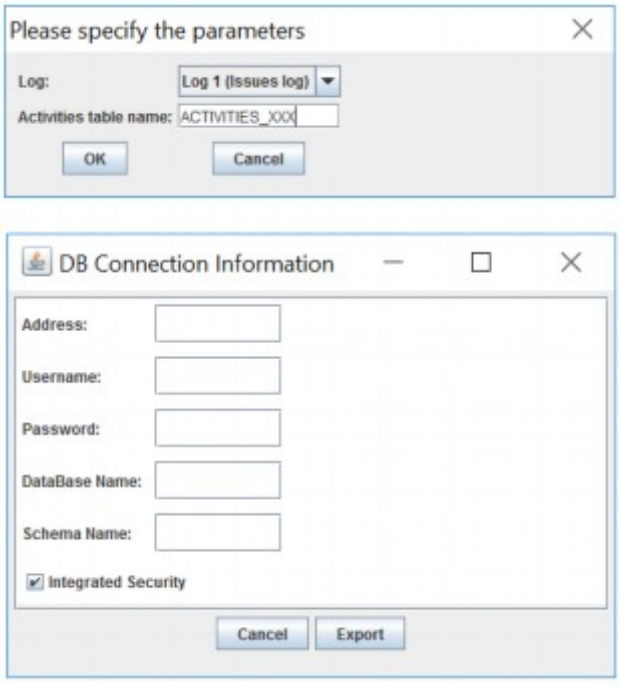


Рис. 3.7. Назва таблиці дій та облікові дані бази даних вилучення

Екстрактор автоматично створить базову таблицю активності, дотримуючись алгоритму, зазначеного в псевдокоді лістингу 3.5. Він взаємодіє з базою даних зіставлення журналу подій. Усі запити створюються без участі користувача.

Лістинг 3.5. Базовий псевдокод генерації журналу подій

```
def GenerateSimpleEventLog():

  var   dataBaseInstance    #Event Log Mapping Database Instance
  var   selectedLog         #Selected event log
  var   currentTrace        #Current trace
  var   currentEvent        #Current event
  var   currentCaseId       #Current case id
  var   currentEventName    #Current event name
  var   currentTimestamp    #Current timestamp
  var   eventLogRow         #Row consisting of caseId, eventName, timestamp
  var   simpleEventLog      #List of eventLogRow

  for currentTrace in selectedLog:
      currentCaseId = currentTrace.getCaseId()
      for currentEvent in currentTrace:
          currentEventName = currentEvent.getName()
          currentTimestamp = currentEvent.getTimestamp()
          eventLogRow.add(currentCaseId, currentEventName, currentTimestamp)
          simpleEventLog.add(eventLogRow)

  return simpleEventLog
```

Користувач може додавати додаткові атрибути до таблиці активностей, вибираючи їх зі списку, отриманого з Бази даних зіставлення журналу подій. Цей список містить всі атрибути з усіх подій. Будь-який атрибут, вибраний з цього списку, буде додано до таблиці активностей як новий стовпець. У випадку, якщо атрибут не включено до певної активності, буде включено нульове значення.

У таблиці 3.3 представлені нові кроки, етапи модифікації та вирішені функції.

Таблиця 3.3.

Опис нових кроків підходу, орієнтованого на артефакти

New Step	Original Step	Modification Stage	Solution for	ID
Data Preparation	-	Replication Database Stage	Ignore non-needed information.	B
			Activities discovery.	K
			Activities granularity discovery.	S
Manual DB Schema Identification*	Database Schema Identification	Replication Database Stage	Database schema identification	E
Artifact Manipulation*	Artifacts Schema Identification	Artifacts Stage	Removal of non-needed attributes.	N
			Removal of non-needed activities.	O
			Removal of repeated activities.	P
			Activity Naming.	Q
			Addition of static attributes.	R
			Activities grouping.	T
Export	-	Event Log Mapping Stage	Activities table generation.	U

Кроки, що містять "*", включені до одного з оригінальних кроків. У цьому розділі ми оцінили екстрактор та виявили відсутні функції, необхідні

для отримання журналів подій для хмарних систем. Крім того, ми представили алгоритми проектування та реалізації, включені до пропонованого екстрактора. Нові та оригінальні кроки в артефактно-орієнтованому підході можна побачити на рисунку 3.8.

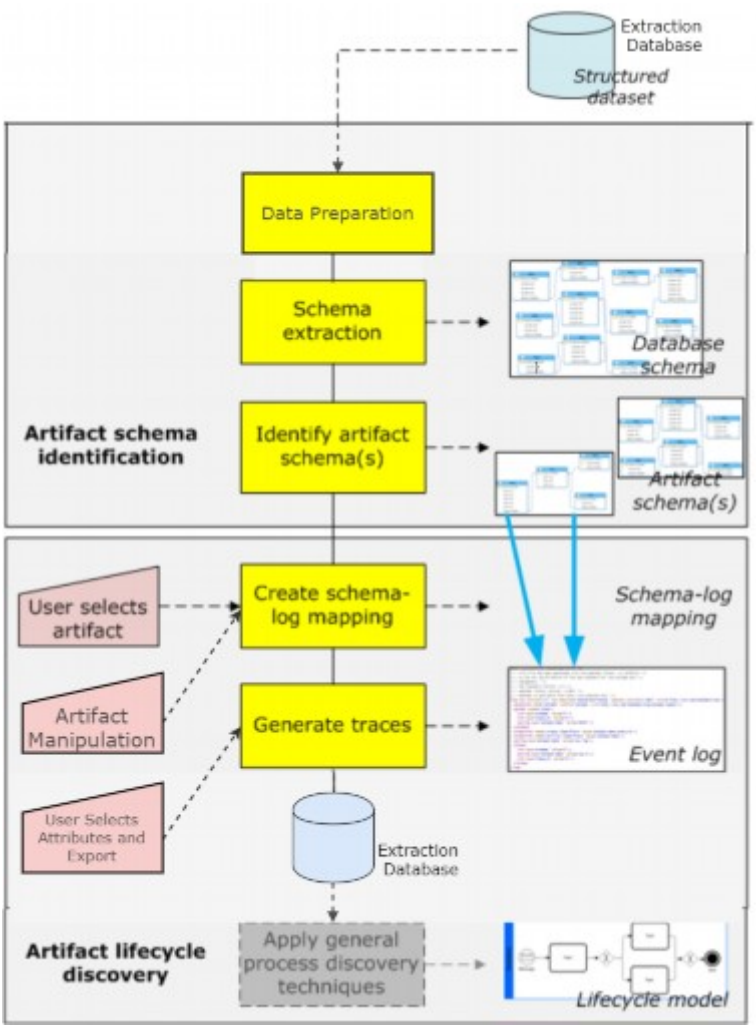


Рис. 3.8. Вдосконалений метод артефакто-орієнтованого підходу

3.4. Представлення отримання даних з хмарних систем за допомогою пропонованого рішення

Метою цього розділу є представити цілі оцінювання. Цілі оцінювання визначаються такими запитаннями: 1) Чи успішно екстрактор API отримує необхідну інформацію? 2) Чи дає традиційний підхід очікувані результати для хмарних систем?, 3) Чи працює орієнтований на артефакти підхід

належним чином і як порівнюють витрачені зусилля з традиційним підходом?

Jira — це власний продукт для відстеження проблем, розроблений Atlassian Corporation. Він забезпечує відстеження помилок, відстеження проблем і функції керування проектами. Цільовою системою для прикладу буде Jira Software. Вибраний процес називається «Керування проблемами». На рисунку 3.9 показано витягнуту реляційну базу даних. Деталі схеми даних опущено.

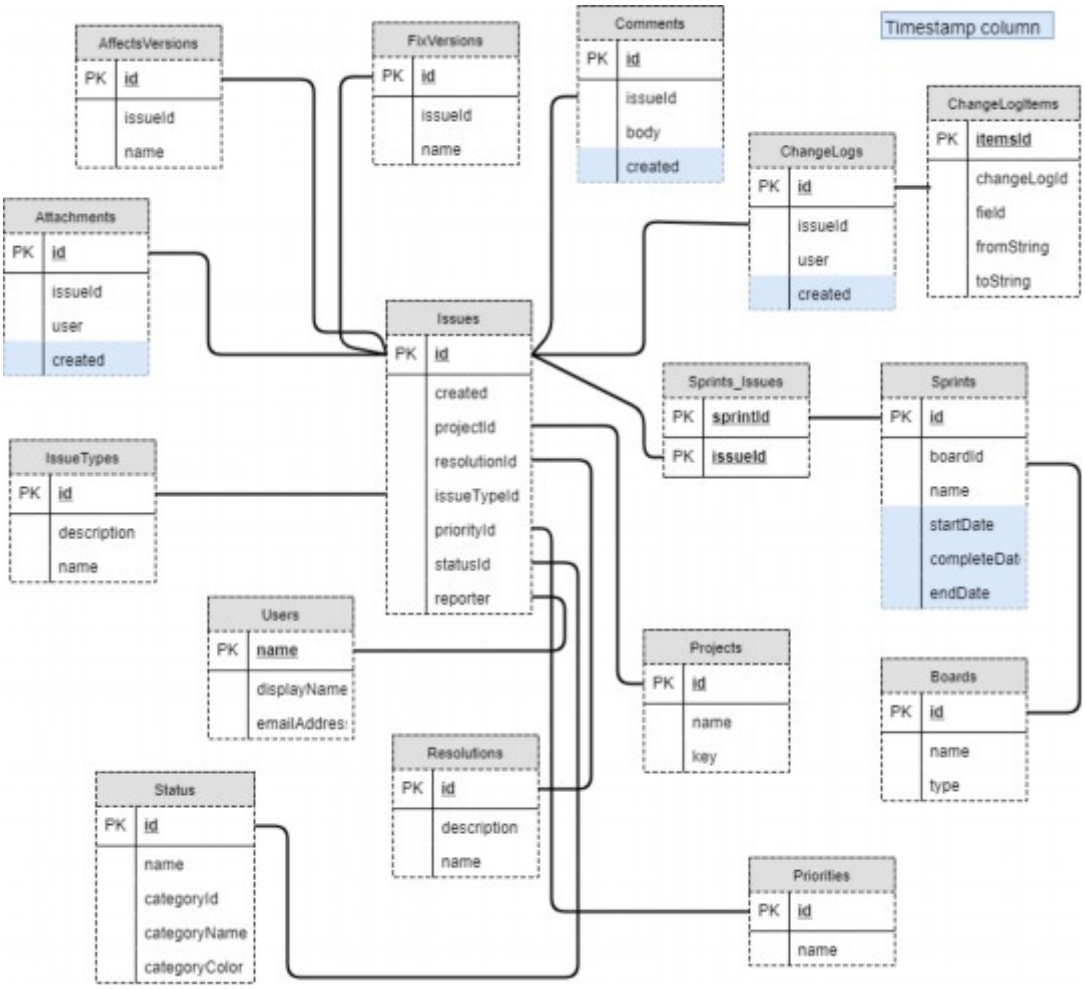


Рис. 3.9. Отримана модель бази даних з Jira

Отримана інформація є повною в тому сенсі, що включено всю відповідну інформацію. Витягнуту інформацію було перевірено шляхом перегляду інтерфейсу Jira. Було проведено детальний аналіз конкретних

проблем, щоб порівняти інформацію, що відображається в інтерфейсі, і витягнуту інформацію. У першому прикладі створена реляційна база даних складається з 16 таблиць із використанням 28 МБ пам’яті.

3.4.1. Використання традиційного підходу

Наразі ми підтвердили, що отримані дані є повними та точними, тому API Extractor виконав свою мету. Коли дані потраплять у базу даних вилучення, починається фаза трансформації. Наступним завданням є створення журналів подій. Процедура, визначена в першому розділі, використовувалася для створення сценаріїв SQL. Для Jira журнал подій містив 1459 випадків, а кількість подій у всіх випадках разом становить 9279.

3.4.2. Застосування артефакто-орієнтованого підходу

Під час фази підготовки даних виконуються усі операції розділення. На рисунку 3.10 показано інтерфейс для вибору таблиць, полів, стовпців із і до стовпців. Окрім обох таблиць змін, слід вибрати ключове відношення між таблицями.

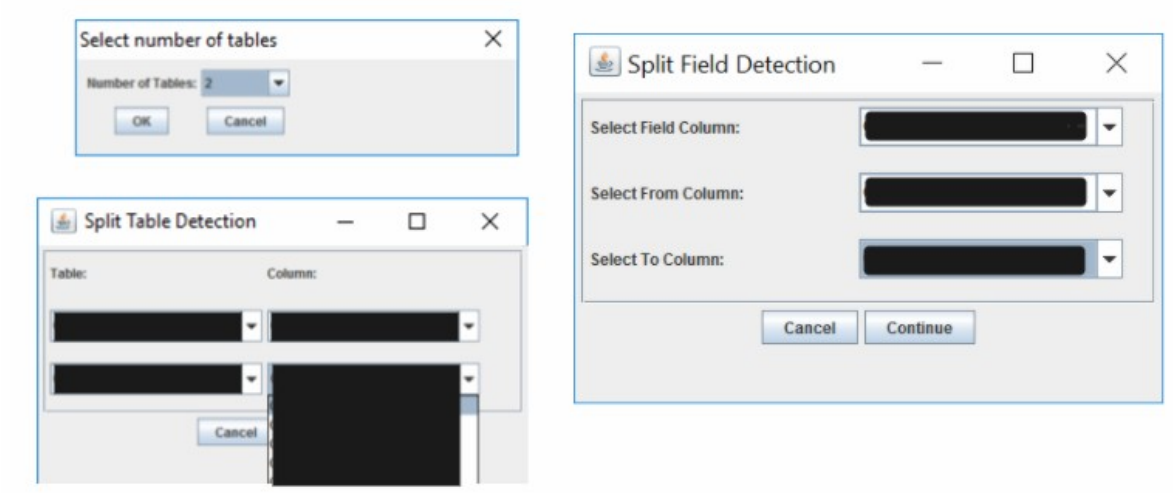


Рис. 3.10. Вибір таблиць та ключів

Після правильного вибору значень будуть показані можливі дії, які містяться в журналі змін. На цьому кроці слід вибрати лише необхідні дії.

Крім того, можна визначити рівень деталізації. На рисунку 3.11 показано вікно вибору. Після того, як таблиці розділено, на етапі підготовки даних не потрібно застосовувати подальші зміни.

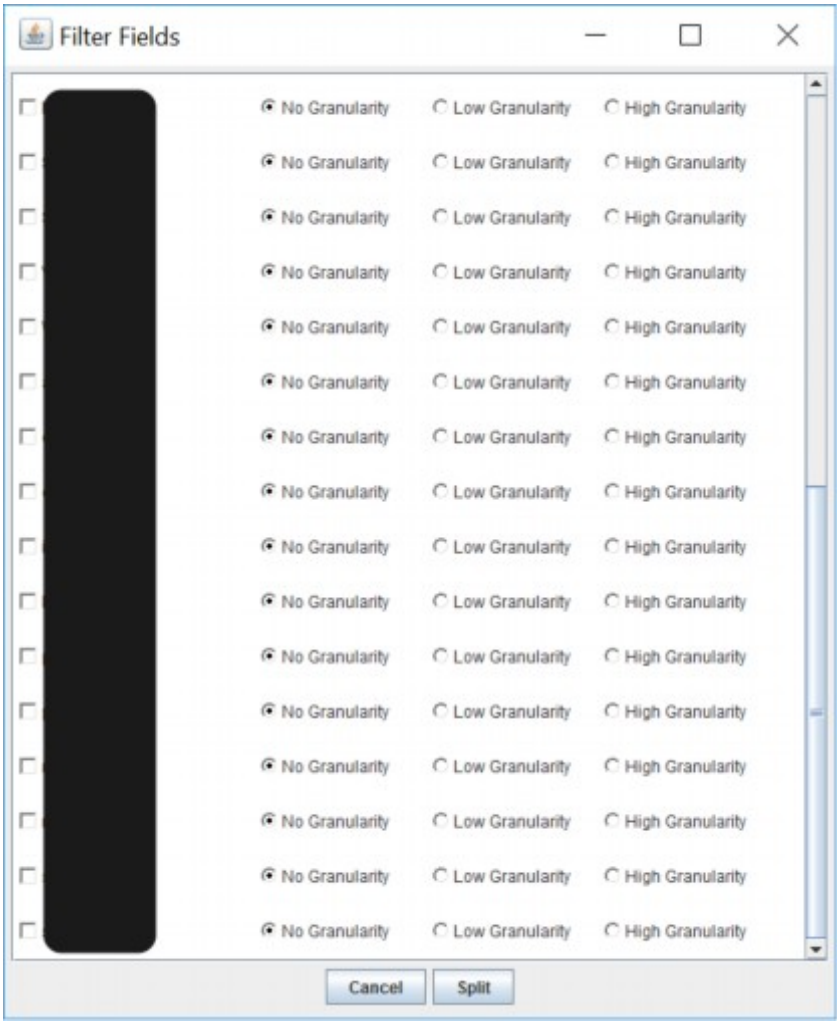
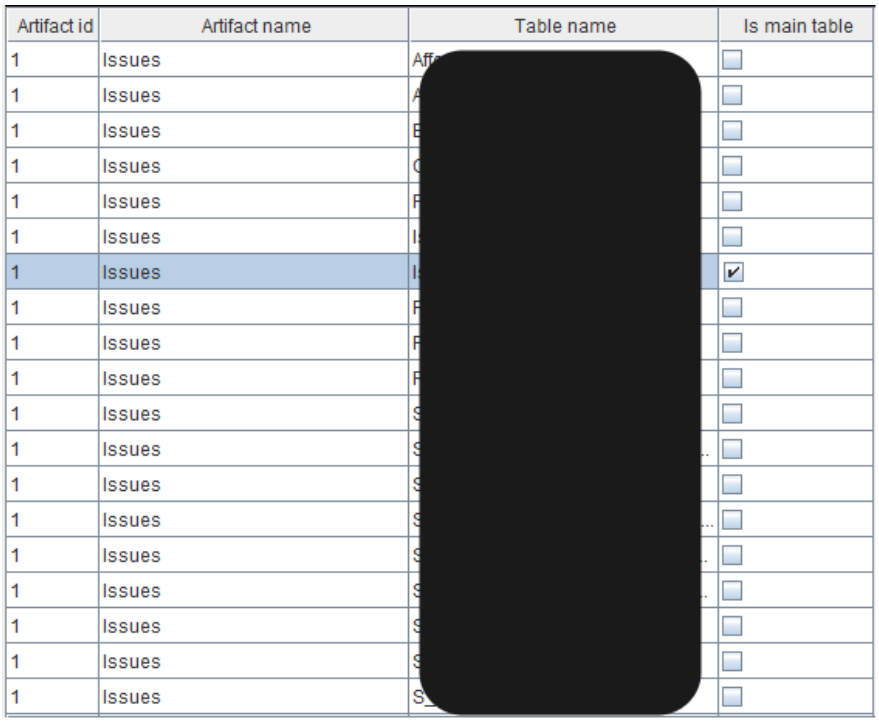


Рис. 3.11. Вибір рівня гранулярності

Першим кроком є ідентифікація домену, яка змогла знайти всі типи значень. Після виявлення доменів необхідно визначити первинний і зовнішній ключі.

Мета кроку (Ідентифікація схеми артефакту) — об’єднати всі таблиці в один артефакт і використовувати Issues як основну таблицю. Тут можна використовувати як ручну специфікацію артефактів, так і автоматичне виявлення через кластеризацію таблиць. Артефакт і його таблиці можна побачити в на рисунку 3.12.



Artifact id	Artifact name	Table name	Is main table
1	Issues	[redacted]	☐
1	Issues		☐
1	Issues		☐
1	Issues		☐
1	Issues		☐
1	Issues		☐
1	Issues		☑
1	Issues		☐
1	Issues		☐
1	Issues		☐
1	Issues		☐
1	Issues		☐
1	Issues		☐
1	Issues		☐
1	Issues		☐
1	Issues		☐
1	Issues		☐
1	Issues		☐
1	Issues		☐

Рис. 3.12. Результати виявлення артефакту

Після виявлення артефакту, його основної таблиці та додаткових таблиць можна ідентифікувати всі атрибути та типи подій. Після ідентифікації цих елементів їх можна видалити.

Крім того, дії можна перейменувати на цьому кроці. Дії, які містять літеру «S» у назві таблиці, були створені з таблиць змін. Імена за замовчуванням враховують лише назви таблиць і стовпців.

Створення схеми для відображення та трасування журналу дуже просте, оскільки не потрібно приймати важливих рішень. Кінцевим результатом цього кроку є файл XES, який містить усі трасування.

Під час кроку експортування створюється таблиця, що містить основний журнал подій. Алгоритм, представлений у цьому розділі, використовується на цьому етапі процедури. Окрім ресурсів, необхідно вибрати атрибут сортування. На рисунку 3.13 показано інтерфейс вибору.

Таблицю діяльності можна перенести в будь-яку базу даних. Як було пояснено в нашому технічному налаштуванні (розділ 3.1) , ця таблиця має бути розташована в базі даних вилучення. Для функції експорту потрібні облікові дані цільової бази даних, щоб перенести таблицю дій.

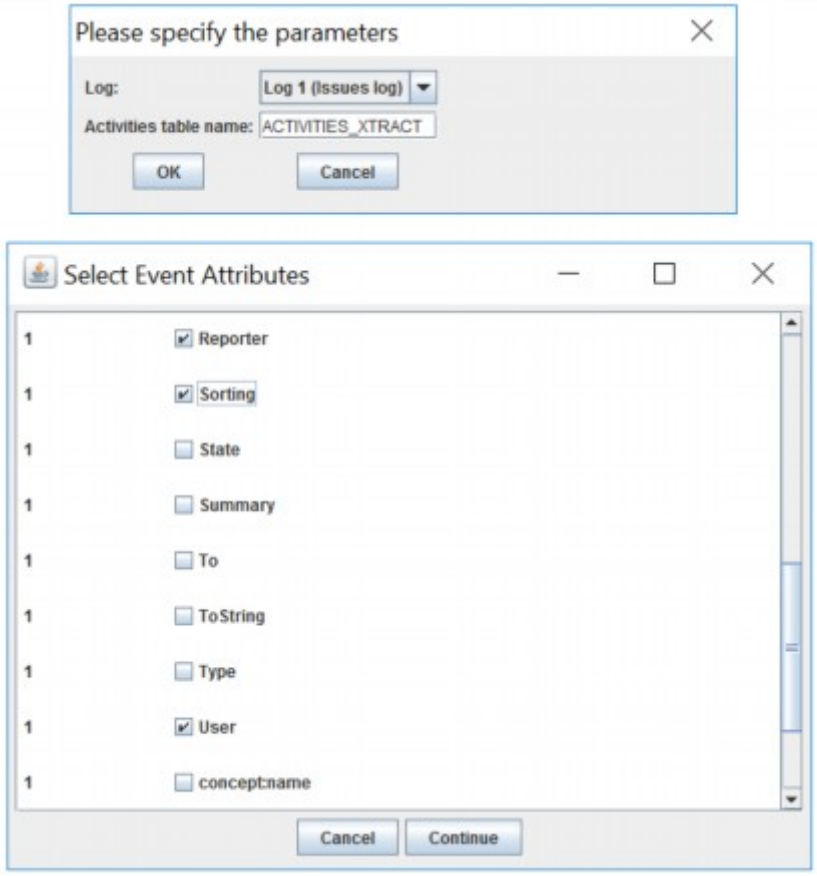


Рис. 3.13. Вибір додаткових атрибутів

Артефакто-орієнтований підхід дав ті самі результати, що й традиційний підхід. У випадку Jira було виявлено 1459 випадків і 9279 загальних дій. На цьому фаза трансформації завершується. Наступним кроком є оцінка обох підходів з двох точок зору.

Як було зазначено підхід , орієнтований на артефакти, дав ті самі результати, що й традиційний підхід. При оцінці базового журналу подій різниця полягає лише в порядку дій. Кількість справ і заходів на кейс однакова. Порядок відрізняється в трасах, де дії мають однакову позначку часу. Ця проблема вирішується за допомогою стовпця сортування, щоб вказати, яку дію слід відобразити першою. Таблиці 3.4 і 3.5 показано один слід, отриманий відповідно за допомогою традиційного підходу та артефактоцентричного підходу. Різниця між порядком журналів подій виділена.

Таблиця 3.4.

Результат, отриманий за допомогою традиційного підходу (Простий журнал подій)

Traditional Approach		
CASE_ID	**ACTIVITY_NAME**	**EVENT_TIME**
14401	Create Issue	2016-04-15 10:25:37.000
14401	Change Status to In evaluation	2016-04-20 20:12:38.000
14401	Change Status to Backlog	2016-08-05 18:40:28.000
14401	Change Issue Type	2016-08-05 18:43:46.000
14401	Change Status to In evaluation	2016-08-05 18:44:28.000
14401	Change Status to Backlog	2016-08-09 22:27:58.000
14401	Change Status to Hold	2016-08-09 22:28:51.000
14401	***Change Status to Declined***	***2016-08-25 20:26:59.000***
14401	***Change Content***	***2016-08-25 20:26:59.000***
14401	Change Issue Type	2016-11-15 15:49:16.000

Таблиця 3.5.

Результат, отриманий за допомогою підходу, орієнтованого на артефакти (простий журнал подій)

Artifact-Centric Approach		
CASE_ID	**ACTIVITY_NAME**	**EVENT_TIME**
14401	Create Issue	2016-04-15 10:25:37.000
14401	Change Status to In evaluation	2016-04-20 20:12:38.000
14401	Change Status to Backlog	2016-08-05 18:40:28.000
14401	Change Issue Type	2016-08-05 18:43:46.000
14401	Change Status to In evaluation	2016-08-05 18:44:28.000
14401	Change Status to Backlog	2016-08-09 22:27:58.000
14401	Change Status to Hold	2016-08-09 22:28:51.000
14401	***Change Content***	***2016-08-25 20:26:59.000***
14401	***Change Status to Declined***	***2016-08-25 20:26:59.000***
14401	Change Issue Type	2016-11-15 15:49:16.000

Ключова відмінність між традиційним підходом і орієнтованим на артефакти підходом полягає в тому, що перший базується на запитах SQL. Це означає, що людина з розширеними знаннями SQL повинна створювати сценарії. Головна перевага підходу, орієнтованого на артефакти, полягає в тому, що SQL-запити створюються автоматично та виконуються внутрішньо. Отже, користувач без розширених знань SQL може успішно використовувати пропонований екстрактор.

У цьому розділі були представлені всі неавтоматичні кроки підходу, орієнтованого на артефакти. Для більшості з них потрібні лише певні параметри. Крім того, було узагальнено відмінності між традиційним підходом і підходом, орієнтованим на артефакти. Найважливішою перевагою артефакто-орієнтованого підходу є автоматична генерація SQL-запитів.

Висновки до розділу

Третій розділ присвячений імплементації методів та алгоритмів для видобування даних із журналу подій хмарних систем. Окреслено ключові компоненти, необхідні для створення системи збору та обробки даних журналу подій у хмарному середовищі. Це включає складові для забезпечення ефективної взаємодії між різними модулями системи, що забезпечує узгодженість та надійність процесів видобування даних. Описано необхідні функції на етапі вилучення даних, що включають механізми для збирання подій, фільтрації та початкової обробки даних. Такі функції є основою для забезпечення точності та повноти інформації, яка потрапляє в журнал подій.

Детально описано методику отримання даних з журналу подій на основі артефактно-орієнтованого підходу. Розглянуто підготовку даних, що передбачає нормалізацію, очищення та структурування інформації для подальшого аналізу. Особливу увагу приділено ідентифікації схеми бази даних, що дозволяє формувати логічні зв’язки між подіями та забезпечувати структурованість у представленні даних.

Представлено реалізацію методики отримання даних із хмарних систем, яка поєднує традиційний підхід та артефактно-орієнтований підхід. Використання традиційного підходу забезпечує базову обробку подій, тоді як артефактно-орієнтований підхід додає гнучкість і точність у відображенні складних процесів, що мають місце у хмарних середовищах.

Розділ демонструє, як можна поєднувати різні підходи та інструменти для ефективного видобування і обробки подій у хмарних системах. Викладені методи сприяють покращенню продуктивності процесу аналізу, забезпечують структурованість отриманих даних і дозволяють використовувати інформацію для вдосконалення управління процесами у хмарних середовищах.

ВИСНОВКИ

У магістерській роботі досліджено методи та підходи до обробки неоднорідних даних журналів подій у хмарних системах. Проведене дослідження охоплює три основні розділи, в яких розглянуто предметну область, вимоги до обробки даних та їх видобування, а також реалізацію методів для побудови ефективної системи журналу подій.

Визначено важливість обробки неоднорідних даних у хмарних системах, які генерують великі обсяги подій, що потребують спеціалізованих підходів для аналізу. Розглянуто інтелектуальний аналіз процесів (Process Mining), який дозволяє отримати корисну інформацію для вдосконалення процесів у хмарних середовищах. Проаналізовано традиційний підхід обробки подій та артефактно-орієнтований підхід, що є гнучкішим для складних структур даних. Також проведено огляд існуючих рішень, зокрема платформи Xesame, метамоделі OpenSlex та онтологічного підходу.

Визначено ключові вимоги до побудови журналу подій у хмарних системах, включаючи точність, повноту та структурованість даних. Розглянуто фази процесу майнінгу (вилучення, трансформація та аналіз), що є основою для забезпечення ефективного аналізу подій у хмарних середовищах. Також описано інструменти реалізації, зокрема використання SQL-скриптів та баз даних, що дозволяють здійснювати ефективну обробку та реплікацію подій. Представлено етапи модифікації даних як важливий елемент інтелектуального аналізу процесу, що включає вилучення, реплікацію, створення артефактів та відображення подій у журналі.

Реалізовано основні компоненти системи для видобування та обробки даних, необхідні функції на етапі вилучення, а також алгоритми на основі артефактно-орієнтованого підходу. Детально описано процес підготовки даних, ідентифікацію схеми бази даних, а також реалізацію методики отримання даних з хмарних систем. Поєднання традиційного підходу з

артефактно-орієнтованим забезпечує більшу гнучкість і точність у обробці подій, що є важливим для складних хмарних процесів.

Загалом, у роботі розроблено та запропоновано комплексну методику для обробки та видобування даних журналів подій у хмарних системах. Запропоновані підходи сприяють підвищенню ефективності аналізу та вдосконаленню управління бізнес-процесами в умовах масштабованих хмарних інфраструктур, що підтверджує практичну цінність виконаного дослідження.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Armbrust, M., et al. (2010). "A View of Cloud Computing." Communications of the ACM.
2. Buyya, R., et al. (2013). Mastering Cloud Computing: Foundations and Applications Programming. Morgan Kaufmann.
3. Dean, J., & Ghemawat, S. (2008). "MapReduce: Simplified Data Processing on Large Clusters." Communications of the ACM.
4. Li, X., et al. (2018). "Deep Learning for Log Analysis in Cloud Computing." IEEE Transactions on Knowledge and Data Engineering.
5. Meng, X., & Zhang, Y. (2019). "Machine Learning-Based Log Analysis for Anomaly Detection in Cloud Systems." IEEE Access.
6. Chen, Y., et al. (2017). "Efficient Log-Based Debugging in Distributed Cloud Systems." ACM SIGOPS Operating Systems Review.
7. Lenk, A., et al. (2011). "What’s inside the Cloud? An Architectural Map of the Cloud Landscape." Proceedings of the IEEE.
8. Zhang, Q., et al. (2010). "Cloud Computing: State-of-the-Art and Research Challenges." Journal of Internet Services and Applications.
9. Rakesh Agrawal, Dimitrios Gunopulos, and Frank Leymann. Mining process models from workow logs. In EDBT, 1998.
10. Alfredo Bolt, Massimiliano de Leoni, Wil M. P. van der Aalst, and Pierre Gorissen. Exploiting process cubes, analytic workows and process mining for business process reporting: A case study in education. In SIMPDA, 2015.
11. Alfredo Bolt and Wil M. P. van der Aalst. Multidimensional process mining using process cubes. In BMMDS/EMMSAD, 2015.
12. Al-Dhuraibi, Y., et al. (2018). "Elasticity in Cloud Computing: State of the Art and Research Challenges." IEEE Transactions on Services Computing.
13. Xu, W., et al. (2009). "Detecting Large-Scale System Problems by Mining Console Logs." Proceedings of SOSP.

14. Fisher, K., & Gruber, R. (2005). "PADS: A Domain-Specific Language for Processing Ad Hoc Data." Proceedings of the ACM SIGPLAN.
15. J.C.A.M. Buijs. Mapping Data Sources to XES in a Generic Way. Master's thesis, Eindhoven University of Technology, 2010.
16. Chandola, V., et al. (2009). "Anomaly Detection: A Survey." ACM Computing Surveys.
17. Cuzzocrea, A., et al. (2011). "Big Data in Cloud Computing: Features and Issues." Proceedings of DOLAP.
18. Malik, S., & Huet, F. (2011). "Adaptive Fault Tolerance in Real-Time Cloud Computing." Proceedings of IEEE CLOUD.
19. Oliner, A. J., et al. (2012). "Advances and Challenges in Log Analysis." Communications of the ACM.
20. Keegan, K., et al. (2015). "Using Machine Learning to Build Log Analytics Pipelines." IEEE Transactions on Knowledge and Data Engineering.
21. Khan, M., et al. (2018). "Cloud Monitoring and Maintenance: Challenges and Tools." Journal of Cloud Computing.
22. Lu, J., et al. (2017). "A Survey on Cloud Resource Logging and Monitoring Techniques." ACM Computing Surveys.
23. Kieu, T., & Chieu, T. (2012). "Anomaly Detection in Cloud Computing: A Survey and Framework." IEEE Internet Computing.
24. Zomaya, A., & Sakr, S. (2017). Handbook of Big Data Technologies. Springer.
25. Sun, Y., et al. (2012). "Security and Privacy in Cloud Computing: A Survey." Journal of Parallel and Distributed Computing.
26. Fu, Q., et al. (2009). "Execution Anomaly Detection in Distributed Systems through Log Analysis." IEEE Transactions on Dependable and Secure Computing.
27. Schneider, J., & Borenstein, D. (2001). "The Use of Log Analysis for Cloud Performance Optimization." International Journal of Cloud Computing.

28. Sharma, S., & Sood, S. K. (2013). "A Novel Security Scheme Based on Hybrid Encryption for Cloud Data Logging." Future Generation Computer Systems.
29. Gupta, R., & Bedi, P. (2014). "Log Analysis for Security in Cloud Computing." International Journal of Information Technology.
30. Van der Aalst, W. M. (2011). Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer.
31. Nguyen, T., et al. (2017). "Efficient Log Collection and Processing in Cloud Environments." IEEE Access.
32. Sridhar, S., & Ahmed, M. (2015). "Log Aggregation and Analysis Tools in Cloud Computing: A Review." Journal of Cloud Computing.
33. Liu, J., et al. (2015). "Resource Usage Logging for Fault Detection in Cloud Systems." IEEE Transactions on Cloud Computing.
34. Tsai, W. T., et al. (2011). "A Scalable and Reliable Cloud Log Management System." IEEE International Conference on Cloud Computing.
35. Tian, Y., et al. (2017). "Machine Learning and Log Mining for Cloud System Diagnosis." Proceedings of ACM SIGMETRICS.
36. Sato, T., et al. (2016). "Applying Machine Learning for Efficient Cloud Log Analysis." Proceedings of the IEEE Big Data Conference.
37. Baek, W., & Gracia, J. (2015). "Developing High-Performance Log Management in Cloud Platforms." International Journal of Cloud Computing.
38. Emna Hachicha, Walid Gaaloul, and Zakaria Maamar. Social-based semantic framework for cloud resource management in business processes. 2016 IEEE International Conference on Services Computing (SCC), pages 443{450, 2016.
39. Wil M. P. van der Aalst and Minseok Song. Mining social networks: Uncovering interaction patterns in business processes. In Business Process Management, 2004.

40. Lu, M., et al. (2019). "Design and Implementation of Secure Logging for Cloud Forensics." Journal of Information Security and Applications.
41. Tang, C., & Zhou, Y. (2004). "System Logs and Their Analysis: A Survey." IEEE Transactions on Knowledge and Data Engineering.
42. Shen, X., et al. (2014). "Log-Based Monitoring and Analysis of Cloud Infrastructure." Proceedings of the ACM Symposium on Cloud Computing.
43. Mills, H., et al. (2016). "Data Logging and Monitoring Tools in Cloud Environments: An Overview." IEEE Communications Surveys & Tutorials.
44. Min, Z., & Shi, T. (2019). "Scalable Log Analytics for Cloud Systems Using Apache Spark." IEEE Transactions on Cloud Computing.
45. Lee, Y., et al. (2018). "A Framework for Security Logging in Cloud-Based IoT Systems." Future Generation Computer Systems.
46. Saberi, M., et al. (2019). "Monitoring Cloud Environments with Log-Based Data Analytics." IEEE Transactions on Services Computing.
47. Ziawasch Abedjan and Felix Naumann. Advancing the discovery of unique column combinations.In CIKM, 2011.
48. Arya Adriansyah, Natalia Sidorova, and Boudewijn F. van Dongen. Cost-based tness in conformance checking. In ACSD, 2011
49. Arya Adriansyah, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Towards robust conformance checking. In Business Process Management Workshops, 2010.
50. Diego Calvanese, Marco Montali, Alifah Syamsiyah, and Wil M. P. van der Aalst. Ontologydriven extraction of event logs from relational databases. In Business Process Management Workshops, 2015.
51. Monika Gupta, Allahbaksh Asadullah, Srinivas Padmanabhuni, and Alexander Serebrenik. Reducing user input requests to improve it support ticket resolution process. 2017.