МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

СТРУЗІК Владислав Анатолійович

УДК 004.41

ВДОСКОНАЛЕННЯ ТЕХНОЛОГІЙ ПРОВЕДЕННЯ РЕФАКТОРИНГУ БАЗ ДАНИХ ДЛЯ ІНФОРМАЦІЙНИХ СИСТЕМ

05.13.06 – інформаційні технології

АВТОРЕФЕРАТ
дисертації на здобуття наукового ступеня
кандидата технічних наук

КИЇВ – 2020

Дисертацією є рукопис.

Роботу виконано у Національному університеті харчових технологій Міністерства освіти і науки України, м. Київ.

Науковий керівник Кандидат технічних наук, доцент
ГРИБКОВ Сергій Віталійович
Національний університет харчових технологій, м. Київ, доцент кафедри інформаційних систем, в.о. завідувача кафедри інформатики

Офіційні опоненти: Доктор технічних наук, доцент
ПРОКОПЕНКО Тетяна Олександрівна
Черкаський державний технологічний університет МОН України,
завідувач кафедри інформаційних технологій проектування

Кандидат технічних наук, доцент
МІРОНОВА Вікторія Леонідівна
Київський національний університет імені Тараса Шевченка МОН України,
доцент кафедри прикладних інформаційних систем

Захист відбудеться «02» лютого 2021 року о 13:00 годині на засіданні спеціалізованої вченої ради К 26.058.05 у Національному університеті харчових технологій за адресою, м. Київ, вулиця Володимирська 68, корпус «А», аудиторія А-311.

З дисертацією можна ознайомитися у бібліотеці Національного університету харчових технологій за адресою, м. Київ, вулиця Володимирська 68.

Автореферат розіслано «18» грудня 2020 року.

Вчений секретар
спеціалізованої вченої ради **Л.О. Власенко**

ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

Актуальність теми. Розробка інформаційних систем часто пов’язана з необхідністю інтеграції даних із віддалених інформаційних систем та забезпеченням прийнятного часу виконання запиту користувача. Складовою частиною інформаційних систем управління є бази даних. Тому виникає потреба в модифікації бази даних та міграції даних. Рефакторинг баз даних одна з ключових проблем із якою зіштовхуються всі IT-підприємства, що займаються розробкою та супроводом баз даних інформаційних систем. Основна ідея рефакторингу полягає в тому, щоб при невеликих змінах у програмному коді привести дизайн програми в більш зрозумілий і розширюваний вигляд без додавання нової функціональності. У більшості випадків сам процес рефакторингу ітеративний. Кожна ітерація повинна проводити мінімальні та цілісні зміни в дизайні програми.

Під час створення та підтримки інформаційної системи розробники обирають певну модель життєвого циклу програмного забезпечення. Сьогодні рефакторинг найчастіше використовується в моделі екстремального програмування. При використанні рефакторингу, де початково його місце не визначено, витрачається більше людино-годин, а відповідно компанії несуть фінансові втрати. Для економії часу та фінансів доцільно визначити місце рефакторингу в моделях життєвого циклу програмного забезпечення.

Поняття рефакторинг уперше було введено Вільямом Опдайком та Ральфом Джонсоном у 1990 році. Проте ще раніше, у 1986 році Робертом Арнольдом вже було описано сильні та слабкі сторони реструктуризації програмного забезпечення. У 1991 році Вільям Грісволд описує прийоми реструктуризації, які в подальшому лягли в основу рефакторингу. Першою об’ємною роботою, присвяченою рефакторингу стала дисертація доктора філософії Вільяма Опдайка 1992 року. Першою фундаментальною роботою, в якій було чітко описано рефакторинг стала робота Мартіна Фаулера у співавторстві з Кентом Беком, Еріхом Гамма, Джоном Брантом, Вільямом Опдайком та Ліардоном Робертсом, написана у 1999 році. У 2006 році Скотт Емблер та Прамодкумар Садаладж перенесли принципи рефакторингу на процес розробки баз даних, розробивши значну частину операцій. Крім цього над даною темою працювали Robert Cecil Martin (Роберт Сесіл Мартін), Стефан Фаро, Паскаль Лерми, Стівен Макконнелл, Пшеничний Данил Андрійович, David Scott Bernstein (Девід Скот Бернштейн), Steve Halladay (Стів Халлідей), Girish Suryanarayana (Гіріш Сурянаяна), Ganesh Samarthyam (Генеш Самартям), Tushar Sharma (Тушар Шарма), Джерард Месарош. Але в літературі відсутня інформація про операції рефакторингу, що пов’язані з процесом доступу до даних.

На цей час в наукових роботах вітчизняних та зарубіжних науковців достатньо повно описано рефакторинг коду та загальний підхід проведення рефакторингу баз даних. Але не досліджені питання, що пов’язані з: доступом до даних при адмініструванні баз даних; переліком правил версіонування баз даних.

Рефакторинг використовують для зменшення обсягу технічного боргу, що виникає при модернізації інформаційної системи. При не чітко визначеному місці рефакторингу відбувається додаткові витрати часу на його проведення, що, в свою

чергу, впливає на вартість розробки та супроводу інформаційної системи, яка буде відбувається за кошти компанії-розробника, а не за рахунок замовника. Таким чином, невірно проведений рефакторинг призводить до втрати працездатності інформаційної системи на певний час, втрат коштів компанії-розробника, за рахунок оплати часу на його проведення.

Також важливо контролювати зміни в схемі бази даних. На цей час не існує єдиної загальноприйнятої специфікації версіонування баз даних і зазвичай спосіб версіонування приймається у вигляді маніфестів на рівні компанії або команди розробників. Подібний підхід призводить до необхідності вивчення такого маніфесту для розуміння того, яким чином версія схеми бази даних впливає на зворотну сумісність.

Отже, **актуальною науково-прикладною задачею** є удосконалення технологій проведення рефакторингу баз даних для інформаційних систем за рахунок пошуку та розробки нових підходів та методів проведення рефакторингу баз даних.

Зв'язок роботи з науковими програмами, планами, темами. Дослідження, результати яких викладено в дисертаційній роботі, проводилися в Національному університеті харчових технологій відповідно до державних програм і планів НДР:

- НДР «Дослідження та впровадження інформаційних технологій у галузях харчової промисловості та освіті» (Національний університет харчових технологій, кафедра інформаційних систем, № ДР 0117U003475, 2017-2022 рр.);
- НДР «Математичні методи аналізу комп'ютеризованих систем» (Національний університет харчових технологій, кафедра інформатики, № ДР 0117U003477, 2017-2020 рр.).

Роль автора в цих науково-дослідних роботах, де дисертант є безпосереднім виконавцем, полягає в розробці операцій рефакторингу баз даних, що пов'язані з політикою безпеки при створенні та супроводі інформаційних систем, визначені місця рефакторингу в популярних моделях життєвого циклу програмного забезпечення, а також у створенні специфікації *семантичного версіонування баз даних*.

Мета та завдання дослідження. *Метою* дисертаційного дослідження є підвищення ефективності процесу створення та супроводу інформаційних систем за рахунок вдосконалення проведення рефакторингу баз даних для інформаційних систем.

Досягнення поставленої мети передбачає вирішення таких *завдань*:

- проаналізувати світовий досвід проведення рефакторингу в моделях життєвого циклу програмного забезпечення та визначити місце його застосування;
- розробити операції рефакторингу баз даних для вирішення задач, що пов'язані з політикою безпеки при створенні та супроводі інформаційних систем;
- створити специфікацію *семантичного версіонування баз даних*, що дасть можливість контролювати зміни схеми бази даних та виключити необхідність розробки корпоративних маніфестів;
- провести апробацію запропонованих операцій рефакторингу для визначення їх ефективності на практиці, в реальних умовах.

Об'єктом дослідження є розробка та супровід інформаційних систем.

Предметом дослідження є методи і підходи проведення рефакторингу баз даних.

Методи досліджень. Для розв'язання поставлених задач використовувались наступні методи:

- метод порівняння при дослідженні популярних моделей життєвого циклу програмного забезпечення;
- метод системного аналізу та функціональне моделювання для аналізу процесу розробки програмного забезпечення з метою виявлення місця рефакторингу;
- метод експериментального моделювання для моделювання процесу розробки програмного забезпечення з метою оцінки ефективності застосування рефакторингу;
- методи математичної статистики для обробки та оцінки ефекту застосування рефакторингу;
- метод аналізу та синтезу для виявлення складових елементів інформаційних систем, що не покриті операціями рефакторингу;
- метод аналогії, а саме функціонально-структурної аналогії при створенні специфікації *семантичного версіонування баз даних*;
- метод природного експерименту при впровадженні результатів дослідження у виробництві.

Наукова новизна отриманих результатів. В ході вирішення поставлених задач були отримані такі наукові результати:

1. *Вперше розроблено* категорію *рефакторинг доступу*, що дає змогу впроваджувати зміни, пов’язані з доступом до даних, при адмініструванні баз даних без втрати працездатності інформаційної системи.
2. *Вперше розроблено* специфікацію *семантичного версіонування баз даних*, яка базується на специфікації *семантичного версіонування програмного забезпечення*, надає можливість ефективно маркувати й автоматизовано впроваджувати оновлення бази даних.
3. *Вдосконалено* процеси найчастіше вживаних моделей життєвого циклу програмного забезпечення за рахунок визначення місця проведення рефакторингу.
4. *Дістала подальший розвиток* специфікація *семантичного версіонування програмного забезпечення* для маркування версії схеми бази даних, що на відміну від існуючих підходів маркування версії не потребує створення приватних корпоративних маніфестів.

Практичне значення одержаних результатів полягає в тому, що розроблені операції рефакторингу доступу надають можливість впроваджувати зміни, які пов’язані з політикою безпеки зі збереженням безперебійної роботи інформаційної системи, а також надають ряд переваг при декомпозиції монолітної системи з метою використання сервіс-орієнтованої архітектури. Специфікація *семантичного версіонування баз даних* дозволяє контролювати зміни відповідно до затверджених циклів випуску програмного забезпечення. Визначення місця рефакторингу в рамках найчастіше вживаних моделях життєвого циклу програмного забезпечення надають можливість розробникам регламентовано його впроваджувати.

Результати дисертаційного дослідження впроваджено:

- у Національному університеті харчових технологій при виконанні НДР «Дослідження та впровадження інформаційних технологій у галузях харчової промисловості та освіті», кафедра інформаційних систем, № ДР 0117U003475, 2017-2022 рр. (акт впровадження 30.06.2020);
- у Національному університеті харчових технологій при виконанні НДР «Математичні методи аналізу комп'ютеризованих систем», кафедра інформатики, № ДР 0117U003477, 2017-2020 рр. (акт впровадження 05.06.2020);
- у ТОВ «ІНТЕРНЕТ ІНВЕСТ», а саме при роботі над реєстратором доменних імен Imena.UA, з хостинг-провайдером MiroHost та хостингом доменних зон DNSHosting (акт впровадження 17.09.2020);
- ТОВ «ПЕРША УКРАЇНСЬКА ЛІЗИНГОВА КОМПАНІЯ» при роботі з хостинг-провайдером MiroHost (акт впровадження 10.09.2020);
- у ТОВ «УКРАЇНСЬКІ МАГІСТРАЛЬНІ МЕРЕЖІ» під час роботи над точкою обміну трафіком Giganet (акт впровадження 09.09.2020).

Результати дисертаційного дослідження мають науково-практичне значення. Використання розроблених операцій категорії рефакторинг доступу забезпечує безперервну роботу інформаційної системи під час проведення сервісного обслуговування баз даних. Запропонована специфікація *семантичного версіонування баз даних* дозволяє за номером версії зрозуміти чи є вона зворотно сумісною, завдяки чому клієнти компанії мають можливість автоматично переходити на нові версії бази даних, що значно економить час. Визначення місця рефакторингу в популярних моделях життєвого циклу програмного забезпечення, відмінних від моделі екстремального програмування, забезпечує збереження людино-годин за рахунок цільового застосування рефакторингу.

Особистий внесок здобувача. Усі основні положення й результати дисертаційної роботи, що захищаються, одержані автором самостійно. Роботи [9, 15] виконувались без співавторства. У спільних публікаціях автору належить такі результати: у роботах [1, 11] описано проблеми захисту корпоративних сховищ та баз даних, досліджено методи та системи додаткового захисту корпоративних сховищ та баз даних; у роботах [2, 12, 14] описано рефакторинг баз даних як один із прийомів створення інформаційної системи, розглянуто рефакторинг баз даних та описано основні операції рефакторингу баз даних; у статті [3] вперше запропоновано місце рефакторингу в популярних моделях життєвого циклу програмного забезпечення, що виникли до популяризації рефакторингу; у статті [4] розширено інформацію про категорію рефакторинг доступу, що вперше була описана у роботі [15]; у статті [5] проаналізовано два типи контролю версій, в свою чергу, у роботі [10] вперше запропоновано *семантичне версіонування баз даних*; у роботі [6] описано та проаналізовано етапи розвитку рефакторингу, а також його вплив на процес створення програмних продуктів; у роботі [7] проаналізовано «погані запахи бази даних» та необхідність проведення рефакторингу баз даних; у роботі [8] проведено аналіз корпоративних інформаційних систем; у роботі [13] описано доцільність використання еволюційного підходу побудови баз даних та роль рефакторингу в даному підході.

Апробація результатів дисертації. Результати досліджень та розробок, викладених у дисертаційній роботі доповідались та обговорювались на наукових конференціях та семінарах: Четвертій Міжнародній науково-практичній конференції «Обчислювальний інтелект (результати, проблеми, перспективи)» (Київ-Черкаси, 2017); Міжнародній науково-технічній конференції «Захист інформації й безпека інформаційних систем» (Львів, 2017); XXIV Міжнародній конференції з автоматичного управління (Київ, 2017); IV Міжнародній науково-технічній Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами» (2017); 84 міжнародній науковій конференції молодих учених, аспірантів і студентів «Наукові здобутки молоді – вирішенню проблем харчування людства в XXI столітті» (Київ, 2018); 82-ій міжнародній науковій конференції молодих учених, аспірантів і студентів «Наукові здобутки молоді – вирішенню проблем харчування людства в XXI столітті» (Київ, 2016); Восьмій Міжнародній Науково-Технічній Конференції (2017); Міжнародній науково-технічній конференції студентів, аспірантів та молодих вчених «Комп'ютерні Науки, Інформаційні Технології Та Системи Управління» (Івано-Франківськ, 2019); Міжнародній науково-технічній конференції «Програмовані логічні інтегральні схеми та мікропроцесорна техніка в освіті і виробництві» (2018).

Публікації. За темою дисертації опубліковано 15 друкованих праць, у яких викладено основний зміст виконаних досліджень, з них 6 статей у фахових виданнях, що включені у міжнародні наукометричні бази даних, 9 тез доповідей у матеріалах наукових конференцій.

Структура та обсяг роботи. Дисертаційна робота складається зі вступу, основної частини, що включає 4 розділи, висновків, списку використаних джерел, який містить 103 найменувань. Загальний обсяг роботи складає 169 сторінок. Основна частина містить 113 сторінок, включаючи 44 рисунків і 4 таблиці.

ОСНОВНИЙ ЗМІСТ РОБОТИ

У вступі сформовано мету та завдання дослідження, обґрунтовано актуальність обраної теми роботи, подано методи дослідження. Наведено дані щодо зв'язку теми дисертаційної роботи з науковими програмами та планами кафедри інформаційних систем Національного університету харчових технологій. Зазначено наукову новизну, яка полягає у визначенні місця рефакторингу в основних поширених моделях життєвого циклу програмного забезпечення, у створенні нової категорії рефакторингу – *рефакторинг доступу*, у розробці специфікації *семантичне версіонування бази даних*. Представлено результати апробації матеріалів дисертації та описано практичне значення отриманих результатів. Наведено інформацію щодо структури та об'єму роботи, публікації та особистий внесок автора.

У першому розділі «Аналіз побудови інформаційних систем та ролі рефакторингу в ній» проаналізовано монолітний та мікросервісний шаблони архітектури, проаналізовано основні проблеми побудови та супровід корпоративних інформаційних систем, розглянуто рефакторинг як предмет дослідження. Сформовано цілі наукового дослідження.

При розробці корпоративних інформаційних систем важливо вже на перших етапах обрати шаблон архітектури системи, відповідно до якого буде вестись розробка. В дисертаційному дослідженні проаналізовані найпопулярніші шаблони архітектури, а саме монолітний та мікросервісний шаблони, наведені їх переваги та недоліки, а також зв'язок з базою даних.

Нерідко у разі збільшення функціональності програмного забезпечення в процесі розробки виникає проблема сприйняття програмного коду. Для уникнення таких проблем командою розробників проводиться рефакторинг.

Найвідомішими іменами, з якими асоціюють рефакторинг та рефакторинг баз даних є Мартін Фаулер, Кент Бек, Скотт Емблер та Прамодкумар Садаладж, проте розвиток та популяризація рефакторингу почались набагато раніше, ніж публікація фундаментальних робіт цих авторів. У першому розділі розглянуто та проаналізовано літературу, в якій описано реструктуризацію програмного забезпечення та рефакторинг.

На основі проаналізованих літературних джерел сформульовано актуальну науково-прикладну задачу та завдання дослідження.

У другому розділі «Моделювання проведення рефакторингу в моделях життєвого циклу програмного забезпечення та визначення його місця» проаналізовано рефакторинг коду та баз даних, обґрунтовано необхідність проведення рефакторингу, надано рекомендації щодо вибору необхідної операції рефакторингу, тестування програмних додатків та бази даних, а також впровадження операції рефакторингу, створено функціональну модель процесу розробки інформаційної системи, визначено місце рефакторингу в популярних моделях життєвого циклу програмного забезпечення та описані основні ознаки необхідності проведення рефакторингу.

Основними цілями рефакторингу вважаються зменшення технічного боргу, забезпечення доступності сприйняття програмного коду усіма учасниками команди розробників, підвищення відмовостійкості проекту в цілому, зміна програмного коду для забезпечення гнучкої модернізації в процесі розширення проекту новими функціями та програмними модулями без зміни наявного семантичного значення програмного коду. Існує декілька категорій рефакторингу, в кожну з яких входять операції, що направленні на досягнення спільної мети при перетворенні програмного коду. Розуміння цих категорій важливе для раціонального проведення рефакторингу. Для проведення рефакторингу важливо керуватись підходом, що базується на першочерговому тестуванні. Він дозволяє збільшити шанси знаходження будь-яких порушень в роботі, що є наслідком проведення рефакторингу. Потрібно внести зміни в схему бази даних, у випадку необхідності перемістити всі початкові дані, на які можуть впливати зміни, а потім оновити всі зовнішні програми, що мають доступ до схеми. Вся робота має бути організована на базі системи керування версіями. Важливо, після проведення рефакторингу у виробничому середовищі, повідомити про це всім розробникам команди, а також всім представникам зовнішніх організацій, яким буде корисна інформація про внесені зміни.

Поточну версію інформаційну систему S_v представимо у вигляді множини характеристик:

$$S_v\{pm, db\{t, tr, r, w, sp, d, per\}, f\} \quad (1)$$

де *pm* – множина програмних модулів;

db – множина елементів бази даних, що включає в себе множину таблиць *t*, множину тригерів *tr*, множину зв'язків між таблицями *r*, множину представлень *w*, множину збережуваних процедур та функцій *sp*, множину даних які зберігаються у базі даних *d*, множину користувачів та прав доступу *per*;

f – множина функцій системи;

v – версія системи.

Формалізований процес повного рефакторингу представимо наступним відображенням:

$$S_v\{\dots\} \xrightarrow{\text{рефакторинг}} S_{vn}\{\dots\} \quad (2)$$

де $S_{vn}\{\dots\}$ система після проведення рефакторингу з номером версії *vn*.

Процес проведення рефакторингу зображено на рисунку 2.1.

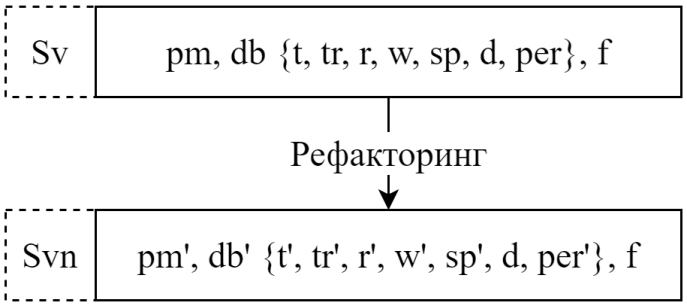

Рис. 2.1. Процес повного рефакторингу

Так як рефакторинг коду полягає в зміні програмного коду без зміни його семантичного значення з метою покращення читання та розуміння програмного коду розробником програмного забезпечення, представимо його наступним відображенням:

$$S_v\{\dots\} \xrightarrow{\text{рефакторинг програмного коду}} S_{vn}\{\dots\} \quad (3)$$

де $S_{vn}\{pm', db\{t, tr, r, w, sp, d, per\}, f\}$ – система після проведення рефакторингу коду;

pm' – множина програмних модулів з впровадженим доповненням, що має в собі проведення рефакторингу коду;

db – множина елементів бази даних, що включає в себе множину таблиць *t*, множину тригерів *tr*, множину зв'язків між таблицями *r*, множину представлень *w*, множину збережуваних процедур та функцій *sp*, множину даних які зберігаються у базі даних *d*, множину користувачів та прав доступу *per*;

f – множина функцій системи, яка ні в якому разі не може бути змінена, адже це суперечить визначенню рефакторингу;

v – початкова версія системи;

vn – версія системи, що відображає зміни після впровадження доповнення, що має в собі проведення рефакторингу.

В свою чергу, рефакторинг бази даних – це зміна схеми бази даних з метою покращення її проекту, яка не призводить до зміни поведінкової та інформаційної семантики бази даних. Рефакторинг бази даних представимо наступним відображенням:

$$S_v\{\dots\} \xrightarrow{\text{рефакторинг бази даних}} S_{vn}\{\dots\} \quad (4)$$

де $S_{vn}\{pm', db'\{t', tr', r', w', sp', d, per'\}, f\}$ – система після проведення рефакторингу бази даних;

pm' – множина програмних модулів з впровадженим доповненням, що має в собі проведення рефакторингу коду, адже впровадження змін до множини елементів бази даних може викликати впровадження змін до множини програмних модулів, через залежність програмного коду від схеми бази даних;

db' – множина елементів бази даних з впровадженим доповненням, що має в собі проведення рефакторингу, в процесі проведення рефакторингу бази даних ні в якому разі не змінюються дані в ній, а відбувається зміна одного чи декількох її елементів, а саме: множини таблиць *t'*, якщо зміни таблиць не було, маємо *t'=t*; множини тригерів *tr'*, якщо зміни тригерів не було, маємо *tr'=tr*; множини зв'язків між таблицями *r'*, якщо зміни зв'язків не було, маємо *r'=r*; множини представлень *w'*, якщо зміни представлень не було, маємо *w'=w*; множини збережуваних процедур та функцій *sp'*, якщо зміни збережуваних процедур та функцій не було, маємо *sp'=sp*, множини користувачів та прав доступу *per'*, якщо зміни користувачів та прав не було, маємо *per'=per*;

f – множина функцій системи, що залишилась незмінною;

v – початкова версія системи;

vn – версія системи, що відображає зміни після впровадження доповнення, що має в собі проведення рефакторингу.

Внесення змін в інформаційну систему можна описати наступним чином:

$$S_v + P_i \rightarrow S_{vn} \quad (5)$$

де S_v – поточна версія інформаційної системи, P_i – доповнення, S_{vn} – система з впровадженим доповненням.

Поточна версія інформаційної системи S_v - це набір множини програмних модулів *pm*, множини елементів бази даних *db*, множини функцій інформаційної системи *f*:

$$S_v = \{pm, db, f\} \quad (6)$$

Зміна інформаційної системи відбувається за рахунок впровадження доповнення P_i, яке включає в себе множину змінених програмних модулів pm_i, множину змінених елементів бази даних db_i, множину змінених функцій інформаційної системи f_i:

$$P_i = \{pm_i, db_i, f_i\} \quad (7)$$

Базуючись на виразі (5) інформаційна система з впровадженим доповненням доцільно представити як об'єднання поточної версії інформаційної системи та доповнення :

$$S_{vn} = S_v \cup P_i \quad (8)$$

Виходячи з виразу (8), а також враховуючи вирази (7) та (8), маємо:

$$S_{vn} = \{pm \cup pm_i, db \cup db_i, f \cup f_i\} \quad (9)$$

Оскільки *pm'* - це множина програмних модулів з впровадженим доповненням, то:

$$pm' = pm \cup pm_i \quad (10)$$

Оскільки *db'* - це множина елементів бази даних з впровадженим доповненням, то:

$$db' = db \cup db_i \quad (11)$$

Оскільки *f'* - це множина функцій з впровадженим доповненням, то:

$$f' = f \cup f_i \quad (12)$$

Таким чином інформаційну систему з внесеними змінами можемо описати наступним чином:

$$S_{vn} = \{pm', db', f'\} \quad (13)$$

Для того, щоб з'ясувати, чи можна вважати впровадження доповнення рефакторингом, доцільно описати предикат наступного виду:

$$R(P_i) = \left(O_{pm}(P_i) \vee O_{db}(P_i)\right) \wedge \overline{O_f(P_i)} \quad (14)$$

$O_{pm}(P_i)$ – предикат, який дорівнює одиниці, якщо множина змінених програмних модулів не порожня:

$$O_{pm}(P_i) = \begin{cases} 1, & \text{якщо } P_i\{pm_i\} \notin \emptyset \\ 0, & \text{якщо } P_i\{pm_i\} \in \emptyset \end{cases} \quad (15)$$

$O_{db}(P_i)$ – предикат, який дорівнює одиниці, якщо множина змінених елементів бази даних не порожня:

$$O_{db}(P_i) = \begin{cases} 1, & \text{якщо } P_i\{db_i\} \notin \emptyset \\ 0, & \text{якщо } P_i\{db_i\} \in \emptyset \end{cases} \quad (16)$$

$O_f(P_i)$ – предикат, який дорівнює одиниці, якщо множина змінених функцій інформаційної системи не порожня:

$$O_f(P_i) = \begin{cases} 1, & \text{якщо } P_i\{f_i\} \notin \emptyset \\ 0, & \text{якщо } P_i\{f_i\} \in \emptyset \end{cases} \quad (17)$$

Щоб визначити який саме рефакторинг був проведений – рефакторинг програмного коду чи рефакторинг бази даних, – необхідно проаналізувати, в яку саме частині було внесено зміни.

Якщо при впровадженні доповнення множина змінених програмних модулів не порожня, а множини змінених елементів бази даних і змінених функцій порожні, то був проведений рефакторинг програмного коду. На мові предикатної алгебри це можна записати наступним чином:

$$R_{code}(P_i) = O_{pm}(P_i) \wedge \overline{O_{db}(P_i)} \wedge \overline{O_f(P_i)} \quad (18)$$

Варто звернути увагу на те, що впровадження змін до множини елементів бази даних може викликати впровадження змін до множини програмних модулів, через

залежність програмного коду від схеми бази даних. Таким чином, рефакторинг бази даних виконується тоді, коли множина змінених елементів бази даних не порожня і множина змінених функцій порожня. На мові предикатної алгебри це можна записати наступним чином:

$$R_{db}(P_i) = O_{db}(P_i) \wedge \overline{O_f(P_i)} \tag{19}$$

Для визначення того, операції якої категорії рефакторингу баз даних використовувалися при створені доповнення використаємо предикатну алгебру.
Категорія *рефакторинг структури* застосовується, якщо виконується рефакторинг бази даних і метою проведення рефакторингу є зменшення технічного боргу структури бази даних:

$$R_{st}(P_i) = R_{db}(P_i) \wedge O_{st} \tag{20}$$

$$O_{st}(P_i) = \begin{cases} 1, & \text{якщо метою є зменшення технічного} \\ & \text{боргу структури бази даних;} \\ 0, & \text{в іншому випадку.} \end{cases} \tag{21}$$

Категорія *рефакторинг якості даних* застосовується, якщо виконується рефакторинг бази даних і метою проведення рефакторингу є поліпшення якості інформації, що зберігається в базі даних:

$$R_{dq}(P_i) = R_{db}(P_i) \wedge O_{dq} \tag{22}$$

$$O_{dq}(P_i) = \begin{cases} 1, & \text{якщо метою є поліпшення якості інформації,} \\ & \text{що зберігається в базі даних;} \\ 0, & \text{в іншому випадку.} \end{cases} \tag{23}$$

Категорія *рефакторинг посилальної цілісності* застосовується, якщо виконується рефакторинг бази даних і метою проведення рефакторингу є збереження посилальної цілісності бази даних:

$$R_{ir}(P_i) = R_{db}(P_i) \wedge O_{ir} \tag{24}$$

$$O_{ir}(P_i) = \begin{cases} 1, & \text{якщо метою є збереження} \\ & \text{посилальної цілісності бази даних;} \\ 0, & \text{в іншому випадку.} \end{cases} \tag{25}$$

Категорія *рефакторинг архітектури* застосовується, якщо виконується рефакторинг бази даних і метою проведення рефакторингу є поліпшення взаємодії зовнішніх програм з базою даних:

$$R_{arch}(P_i) = R_{db}(P_i) \wedge O_{arch} \tag{26}$$

$$O_{arch}(P_i) = \begin{cases} 1, & \text{якщо метою є покращення архітектури;} \\ 0, & \text{в іншому випадку.} \end{cases} \tag{27}$$

Категорія *рефакторинг методів* застосовується, якщо виконується рефакторинг бази даних і метою проведення рефакторингу є покращення якості методів (збережуваних процедур, збережуваних функцій, тригерів):

$$R_m(P_i) = R_{db}(P_i) \wedge O_m \tag{28}$$

$$O_m(P_i) = \begin{cases} 1, & \text{якщо метою є покращення якості методів;} \\ 0, & \text{в іншому випадку.} \end{cases} \quad (29)$$

Процес розробки програмного продукту розбивається на етапи задля вдосконалення дизайну, менеджменту продукту та проекту. Дані етапи визначені у життєвому циклі програмного забезпечення. Життєвий цикл програмного забезпечення передбачає можливий аналіз попередніх результатів, а також артефактів, які розробляє та вдосконалює команда розробників.

Рефакторинг в сучасних моделях життєвого циклу програмного забезпечення є одним із регламентованих прийомів, проте в моделях життєвого циклу програмного забезпечення, що були створені до популяризації рефакторингу, цей прийом не передбачено. Під час дисертаційного дослідження визначено місце рефакторингу в наступних моделях життєвого циклу програмного забезпечення: каскадна модель, V-модель, інкрементна модель, Rapid Application Development модель, модель екстремальне програмування, ітеративна модель, спіральна модель.

У третьому розділі «Розробка операцій рефакторингу доступу та специфікації семантичного версіонування баз даних» створено нову категорію *рефакторинг доступу* та запропоновано можливий подальший розвиток рефакторингу, а також створено специфікацію *семантичне версіонування баз даних*.

Відповідно до семантичного версіонування, версія програмного продукту містить в собі три числа, розділених крапкою, де кожне з чисел (X.Y.Z) має відповідну назву — мажорна.мінорна.патч. У запропонованій специфікації *семантичне версіонування бази даних* зміни мажорної, мінорної та патч-версії відбуваються у наступних випадках:

- мажорна версія інкрементується при додаванні нового поля до таблиці, при видалені поля із таблиці, при переіменувані поля, при видалені NULL-значення, при додаванні обмежень;
- мінорна версія інкрементується при додаванні нового елементу, при додаванні нового поля, за умовою, що воно буде містити значення за замовчуванням, при додаванні NULL-значення, при видаленні обмежень, при зміні тригерів;
- патч-версія інкрементується при зміні типів даних, за умови, що початковий і кінцевий типи даних є сумісними, при додаванні NULL-значення, якщо це є виправленням помилки проектування бази даних, при видаленні обмежень, якщо це є виправленням помилки проектування.

Сьогодні виділено шість категорій рефакторингу баз даних, відповідно до яких застосовуються певні операції. Проте жодна існуюча категорія не описує зміни, пов’язані з доступом до даних, при адмініструванні системи управління базою даних.

Тому доцільно створити нову категорію – категорія *рефакторинг доступу*, що має в собі операції, які дають можливість контрольовано виконувати зміни у частині обмеження доступу при адмініструванні системи управління базою даних, створювати регламентовані процеси реакції на події, пов’язані з політикою безпеки.

До категорії *рефакторинг доступу* належать наступні операції:

- зміна атрибутів автентифікації;

- звуження привілеїв доступу;
- розширення привілеїв доступу;
- виділення схеми бази даних;
- злиття схем баз даних.

Метою використання операції *зміна атрибутів автентифікації* є зміна способів або атрибутів автентифікації. Основними причинами застосування цієї операції є компрометація пароля користувача або планова ротація паролів.

Процес застосування операції *зміна атрибутів автентифікації* зображено на рисунку 2.

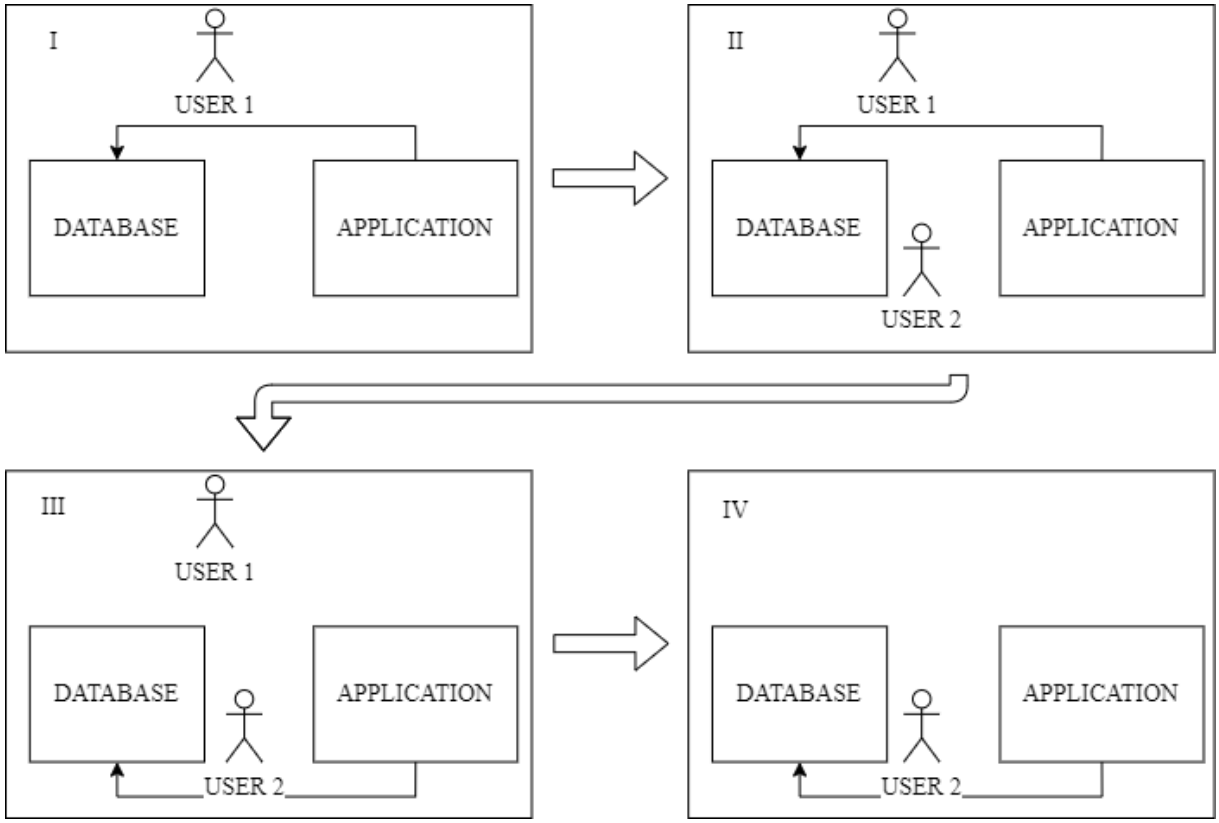

Рис. 2. Процес застосування операції *зміна атрибутів автентифікації*

На рис. 2 відображено чотири стани проведення операції рефакторингу:

I. Існує база даних (Database), що пов’язана з деяким програмним додатком (Application) та користувач (User 1), який має доступ до цієї бази даних.

II. Створити нового користувача (User 2).

III. Змінити користувача бази даних у тестовому та виробничому середовищі. Провести аудит бази даних на предмет наявності автентифікації за атрибутами старого користувача.

IV. Видалити старого користувача (User 1).

Моделюючи ситуацію заміни паролів доступу до бази даних інформаційної системи, припустимо, на одного користувача витрачається T хвилин. Отже, при заміні паролів без використання операції *зміна атрибутів автентифікації* для N користувачів бази даних витрачається $T*N$ хвилин, що буде являти собою період

повної або часткової непрацездатності інформаційної системи. За 1 хвилину обробляється *X* запитів клієнтів до бази даних. Таким чином, за період зміни паролів буде відхилено *T*N*X* запитів клієнтів.

А при використанні операції *зміна атрибутів автентифікації* категорії *рефакторинг доступу* не втрачається працездатність інформаційної системи та всі запити клієнтів будуть успішно оброблені.

Метою впровадження операції *звуження привілеїв доступу* є скасування привілеїв доступу до об'єктів бази даних, що на момент впровадження операції рефакторингу були надані.

Потреба використання даної операції виникає після виділення вузького місця в окрему базу даних, при перетворенні монолітної системи у сервіс-орієнтовану, а також при зміні політики доступу до даних для користувачів системи.

Процес застосування операції *звуження привілеїв доступу* зображено на рисунку 3.

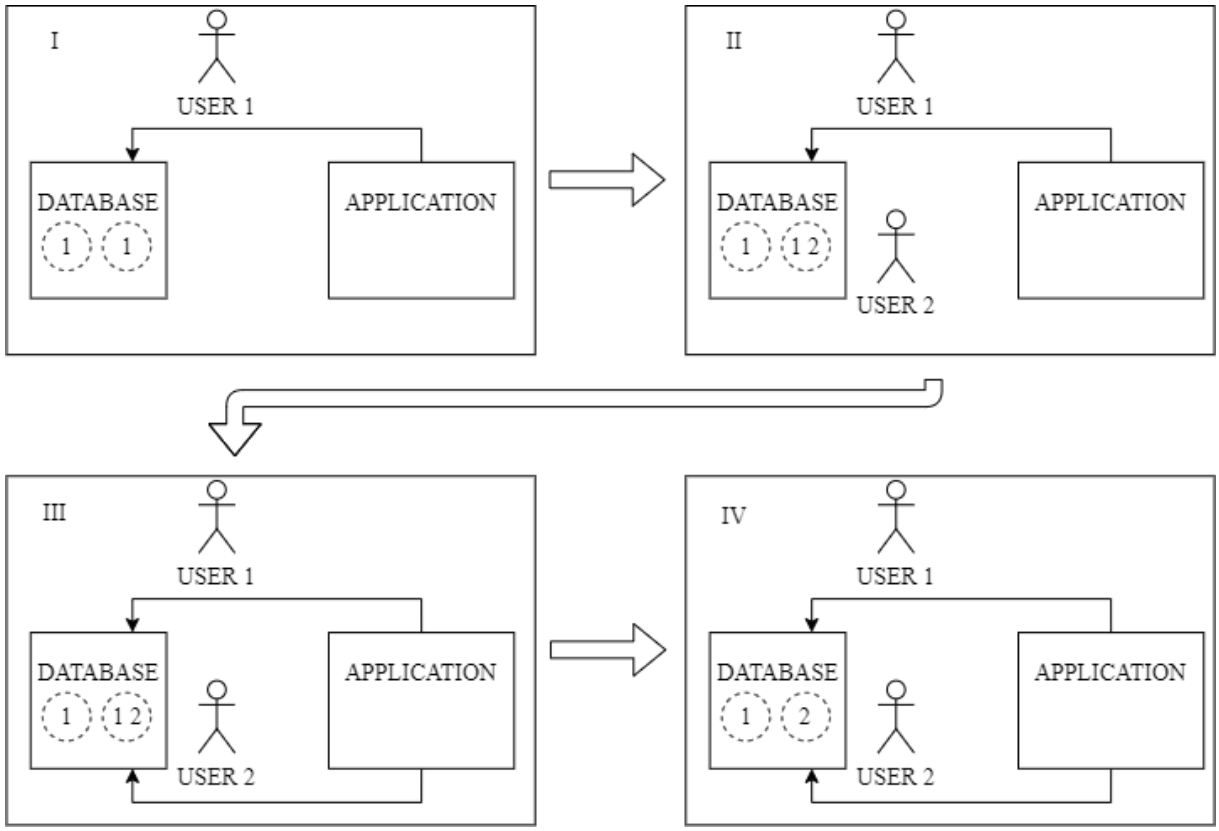

Рис. 3. Процес застосування операції *звуження привілеїв доступу*

На рис.3 відображено чотири стани проведення операції рефакторингу:

I. Існує база даних (Database) з об'єктами, доступ до яких є у певного користувач (User 1). Ця база даних пов'язана деяким програмним додатком (Application).

II. Створити нового користувача (User 2).

III. Новий користувач використовується в місцях звуження привілеїв. Проводиться аудит бази даних на предмет наявності авторизації старого користувача за привілеями, які підпали під звуження.

IV. Видалення зі старого користувача привілеїв, які підпали під звуження.

При звуженні привілеїв доступу до бази даних розробник може зіштовхнутись з тим, що забравши привілеї у певних користувачів залишається частина бази даних, доступ користувачами до якої взагалі відсутній. Тобто на *T* хвилин буде відсутній доступ до певної частини бази даних для *N* користувачів, що виконують *X* запитів до бази даних за 1 хвилину. До моменту відновлення доступу до цієї частини бази даних буде відхилено *T*N*X* запитів.

Тоді, як використання операції *звуження привілеїв доступу* категорії *рефакторинг доступу* дає змогу уникнути такої ситуації. Адже спочатку створюється новий користувач, який використовується в місцях звуження привілеїв. Таким чином всі запити можуть бути оброблені.

Операція *розширення привілеїв доступу* впроваджується для надання існуючому користувачу прав доступу до об’єктів бази даних, що йому недоступні.

Найчастіше використовується після зміни схеми бази даних, що впроваджувалися задля доповнення або додавання до відображення в реляційній базі даних сутностей реального світу.

Процес застосування операції *розширення привілеїв доступу* зображено на рисунку 4.

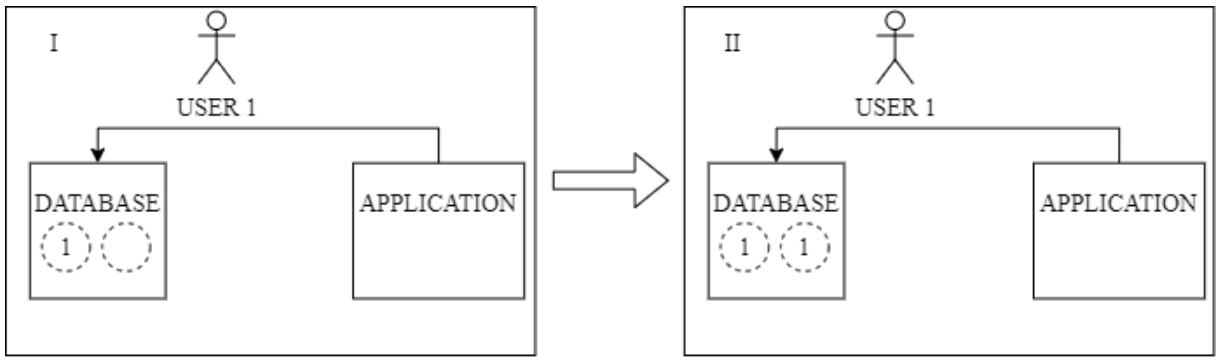

Рис. 4. Процес застосування операції *розширення привілеїв доступу*

На рис. 4 відображено два стани проведення операції рефакторингу:

I. Існує база даних (Database) з об’єктами, доступ до яких необхідно отримати певному користувачу (User 1). Ця база даних пов'язана деяким програмним додатком (Application).

II. Користувачу надають необхідні привілеї доступу.

При розширенні привілеїв доступу не виникає ситуації, за якої база даних стає недоступною. Проте, щоб розширити привілеї доступу розробнику необхідно в першу чергу розробити оптимальну послідовність виконання дій для досягнення бажаного результату. Таким чином, витрачається *PT* людино-годин для розроблення плану дій, що може бути досить різним в залежності від досвіду та кваліфікованості розробника.

Використання операції *розширення привілеїв доступу* категорії *рефакторинг доступу* допомагає зберегти *PT* людино-годин, які можуть бути витрачені на розробку плану дій.

Метою впровадження *виділення схеми бази даних* є виокремлення групи таблиць, представлень та збережуваних процедур, що їх обслуговують (сегменту даних), до окремої бази даних.

Застосування даної операції знаходить своє місце під час переходу між варіантами взаємодії з базою даних в сервіс-орієнтованій архітектурі за потреби подальшого масштабування операцій зчитування, запису або обчислення.

Процес застосування операції *виділення схеми бази даних* зображено на рисунку 5.

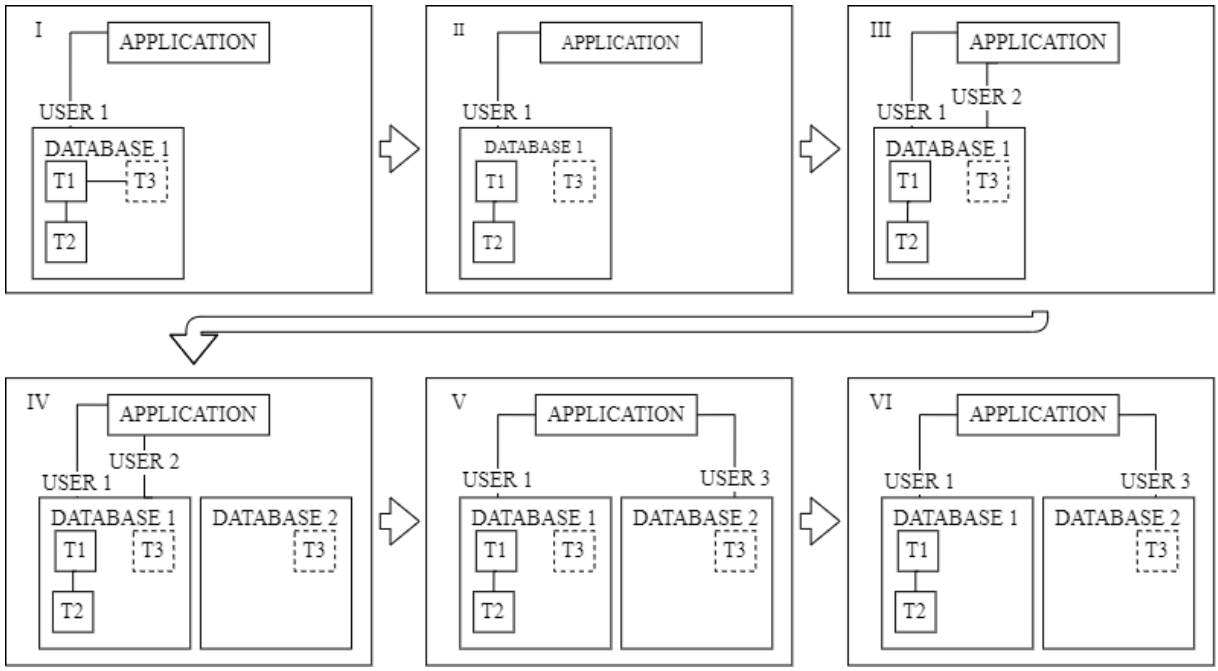

Рис. 5. Процес застосування операції *виділення схеми бази даних*

На рис. 5 відображено шість станів проведення операції рефакторингу:

I. Визначити сегмент даних (T3) в існуючій базі даних (Database 1).

II. Видалити зв’язаності між даними, що переміщуються.

III. Створити нового користувача (User 2) та перевірити працездатність інформаційної системи у тестовому середовищі. Провести аудит бази даних на предмет звернень до виокремлених даних від імені старого користувача.

IV. Створити нову базу даних (Database 2) та виконати міграцію даних.

V. Замінити нового користувача старої бази даних (User 2) на користувача нової бази даних (User 3).

VI. Видалити виокремлений сегменту (T3) зі старої бази даних (Database 1).

Щоб виділити певну частину бази даних в окрему базу даних, необхідно, в першу чергу, обмежити доступ до цієї бази даних. Таким чином, за потреби виділити сегмент бази даних в окрему базу даних, інформаційна система втрачає працездатність на T хвилин, протягом яких N користувачів в загалом виконують $N*X$ запитів до бази даних. Тобто буде відхилено $T*N*X$ запитів.

При використані операції *виділення схеми бази даних* категорії *рефакторинг доступу* працездатність інформаційної системи зберігається, так само як і доступ до

всіх елементів бази даних, зміни в яку вносяться. Таким чином всі запити можуть бути оброблені.

Операція *злиття схем баз даних* націлена на злиття різних баз даних в єдину, з метою спрощення реалізації бізнес-транзакцій та забезпечення консистентності даних тощо.

Можливе застосування даної операцій знаходить своє місце під час переходу від підходу *база даних кожному сервісу* до *спільна бази даних*.

Процес застосування операції *злиття схем баз даних* зображено на рисунку 6.

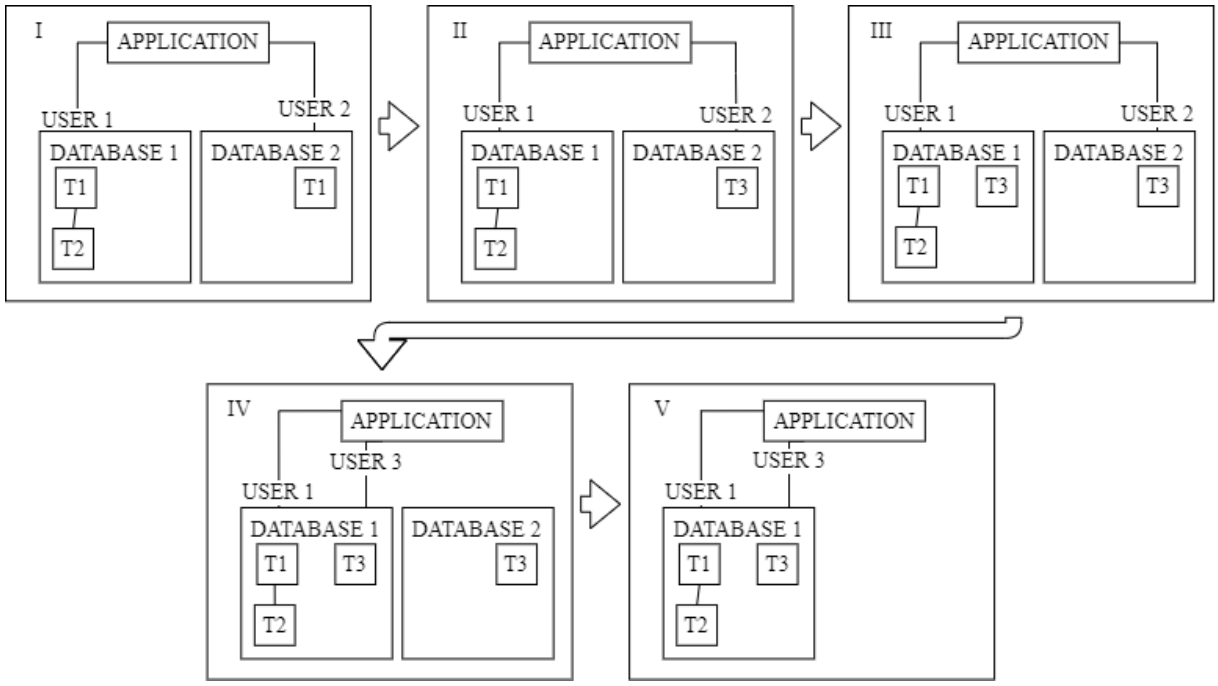

Рис. 6. Процес застосування операції *злиття схем баз даних*

На рис. 6 відображено п’ять станів проведення операції рефакторингу:

I. На початковому етапі є дві бази даних, для яких необхідно виконати злиття (Database 1, Database 2), що пов’язані з певним програмним додатком (Application).

II. Уніфікувати назви об’єктів зливаємих баз даних.

III. Визначити головну базу даних з-поміж зливаємих та провести міграцію даних.

IV. Замінити користувача другорядної бази даних (User 2) на відповідного користувача головної бази даних (User 3). Перевірити працездатність інформаційної системи у тестовому середовищі та провести аудит бази даних на предмет звернень до даних від імені користувачів другорядної бази даних.

V. Видалити другорядну базу даних.

При виникненні бізнес-потреби злиття декількох баз даних, втрачається доступ до кожної із баз даних, що зливаються. На злиття M баз даних сумарно витрачається T хвилин. Так як за 1 хвилину користувачі виконують X запитів до бази даних. Таким чином за T хвилин буде відхилено $T*X$ запитів користувачів.

При використані операції *злиття схеми бази даних* категорії *рефакторинг доступу* працездатність інформаційної системи зберігається, так само як і доступ до всіх елементів баз даних, зміни в які вносяться. Таким чином всі запити можуть бути оброблені.

Використання створеної категорії *рефакторинг доступу* значно зменшує час на сервісне обслуговування та забезпечує зміну налаштувань автентифікації та авторизації без втрати працездатності інформаційної системи.

Аналізуючи схему покриття рефакторингом програмного продукту, яка зображена на рис. 7, зрозуміло, що частина, що відноситься до комп'ютерної мережі рефакторингом не покрита. Тому, можливим наступним етапом у розвитку рефакторингу може стати рефакторинг мережі.

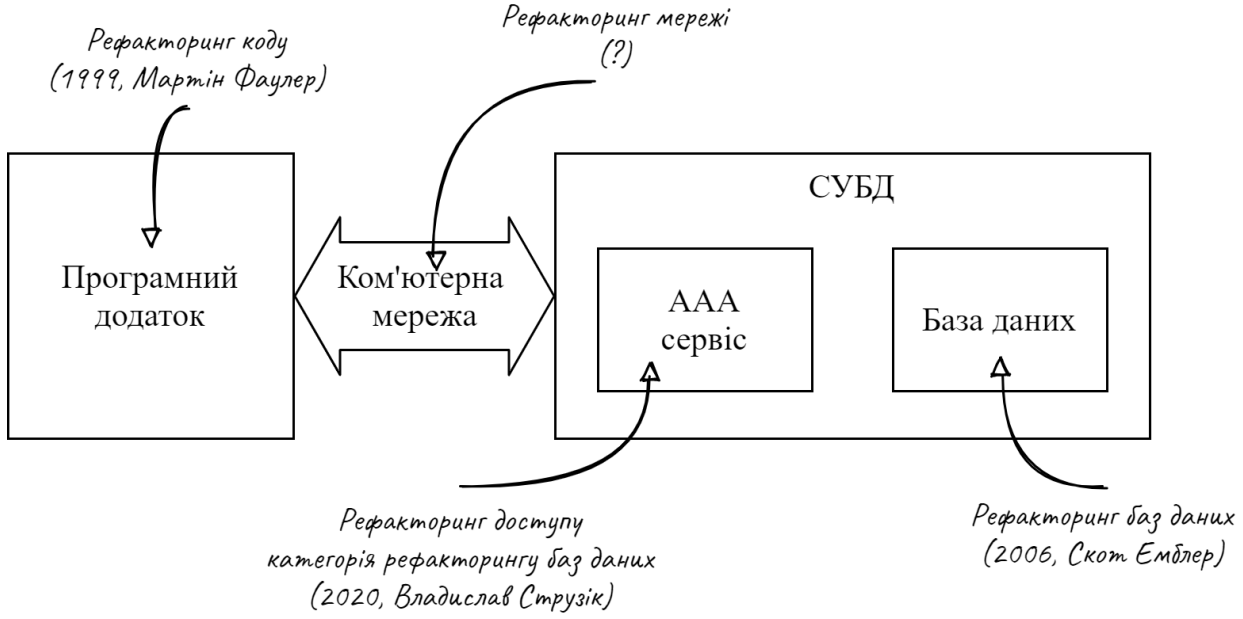

Рис. 7. Покриття програмного продукту рефакторингом

При проведенні змін в програмному коді та базі даних, важливо проводити контроль версій, адже це забезпечує відновлення втрачених файлів, можливість порівняння різних версій та повернення до робочої версії системи, у випадку несправності. *Семантичне версіонування програмного коду* – це процес, що проводиться відповідно до встановленої специфікації. Не дивлячись на те, що бази даних також піддаються змінам під час підтримки програмного продукту, чіткого регламенту щодо версіонування баз даних не існує.

Автором дисертаційного дослідження запропоновано специфікацію *семантичне версіонування баз даних*.

У четвертому розділі «Теоретичне та практичне використання результатів дисертаційного дослідження» проаналізовано результати імітаційного моделювання, а також впровадження в практичну діяльність вдосконалених за рахунок визначення місця рефакторингу в моделях життєвого циклу програмного забезпечення, операцій рефакторингу доступу, специфікації *семантичне версіонування баз даних*.

В ході дисертаційного дослідження розроблено імітаційну модель, з метою визначення необхідності проведення рефакторингу при розробці системи. Модель перевірена на адекватність за допомогою критеріїв Стьюдента та Фішера.

Основними вхідними даними є:

- розмір системи - початкова кількість елементів системи;
- кількість задач - кількість змін, які необхідно впровадити;
- досвідченість команди розробників - кваліфікація команди розробників у відсотках, де 0% - не кваліфікована команда, 100% - висококваліфікована команда.

Моделювання було проведено при різних наборах вхідних даних, з яких основні представлені у таблиці 1.

Таблиця 1. Результати імітаційного моделювання

№ експерименту	Вхідні дані			Результати			
	Розмір системи (ел.)	Кіл-ть задач (шт.)	Досвід команди розробників	Кінцевий технічний борг без рефакторингу	Кінцевий технічний борг з рефакторингом	Загальний час виконання задач без рефакторингу (год.)	Загальний час виконання задач з рефакторингом (год.)
1	1 000	1 000	85%	100%	10%	42129.79	28685.63
2	1 000	500	85%	100%	8%	20418.57	14109.09
3	1 000	100	85%	92%	7%	3726.88	2928.82
4	1 000	1 000	50%	100%	8%	48762.28	36747.09
5	1 000	500	50%	100%	8%	24348.85	18626.15
6	1 000	100	50%	91%	6%	4123.83	3536.7
7	1 000	1 000	15%	100%	12%	56190.85	45465.92
8	1 000	500	15%	100%	8%	27749.76	22728.78
9	1 000	100	15%	93%	11%	5024.83	4551.32

Аналізуючи результати моделювання, виявлено, що проведення рефакторингу сприяє, в першу чергу, зменшенню технічного боргу, що, в свою чергу, призводить до скорочення часу розробки системи та підвищенню її якості.

На рис. 8 відображено графік, побудований за такими вхідними даними: розмір системи - 1000 елементів, кількість задач, які потрібно обробити - 1000, досвід розробників 85%.

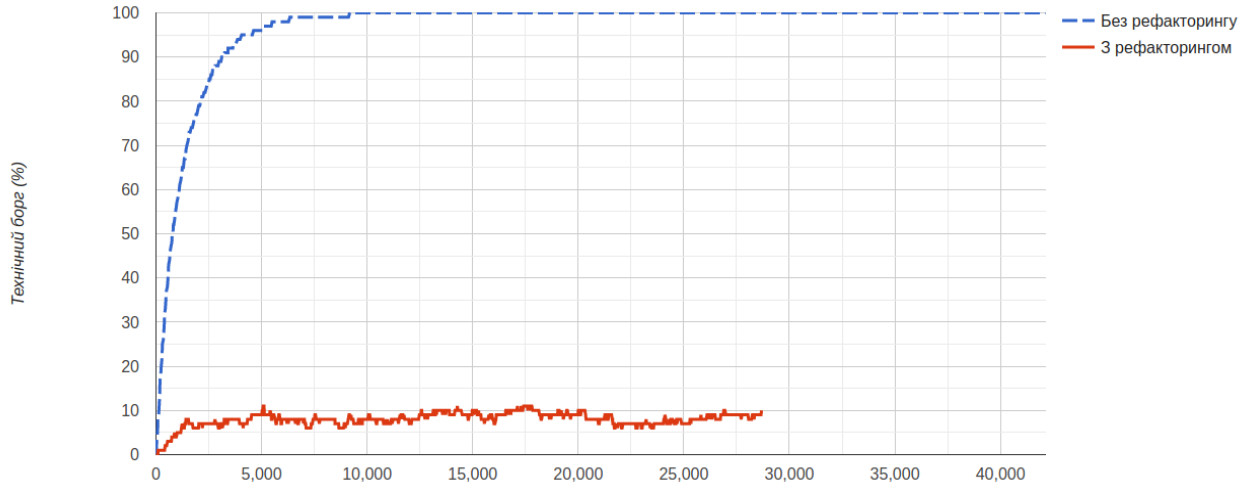

Рис. 8. Графік зростання технічного боргу при розробці висококваліфікованою командою.

На основі отриманих результатів, встановлено, що використання рефакторингу сприяє зменшенню загального часу на проведення розробки та зниженню технічного боргу, відповідно якість кінцевого продукту вища.

Доцільно також проаналізувати дані, які були отримані при таких самих вхідних умовами, але для менш кваліфікованої команди. На рис. 9 відображено графік, побудований за такими вхідними даними: розмір системи - 1000 елементів, кількість задач, які потрібно обробити - 1000, досвід розробників 15%.

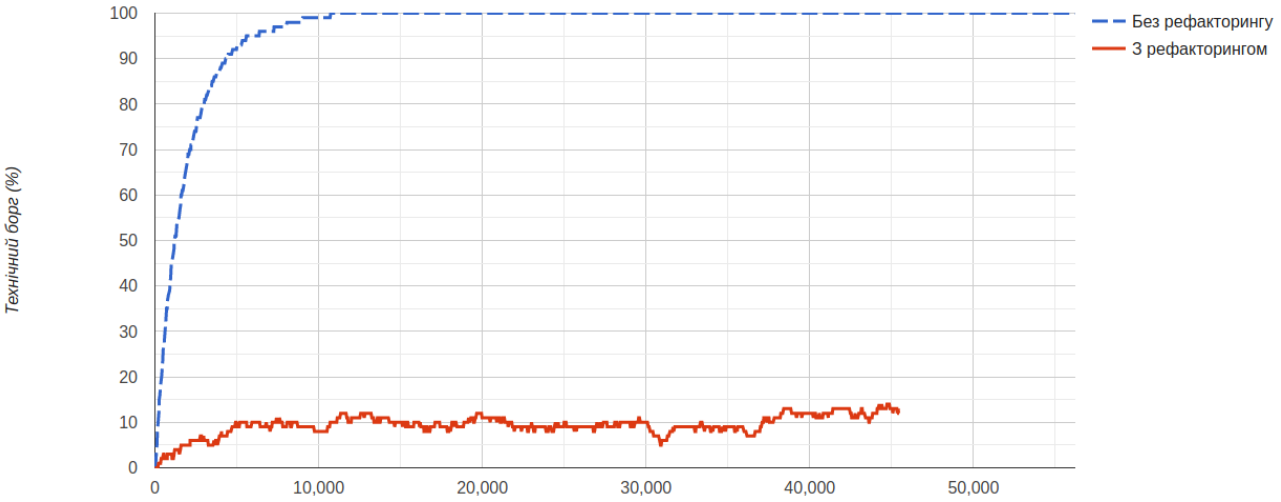

Рис. 9. Графік зростання технічного боргу при розробці низько-кваліфікованою командою.

Використання рефакторингу сприяє підвищенню швидкості розробки системи та якості програмного продукту незалежно від кваліфікації команди розробників. Це підтверджено аналізом результатів імітаційного моделювання, що відображені на рис. 8, 9.

На основі отриманих результатів від розробників програмного забезпечення, які виконали апробацію наукових результатів дисертаційної роботи, встановлено, що використання операцій рефакторингу доступу забезпечує: безперервну роботу інформаційної системи під час впровадження змін, пов’язаних з доступом до даних, при адмініструванні системи управління базою даних; відсутність сервісного вікна для впровадження операцій, які пов’язані з політикою безпеки; відсутність необхідності прийняття рішень при впровадженні змін у частині обмеження доступу при адмініструванні системи управління базою даних.

Запропоновані специфікація *семантичне версіонування баз даних*, нова категорія *рефакторинг доступу* та оновлені, за рахунок визначення місця рефакторингу, моделі життєвого циклу програмного забезпечення пройшли апробацію в українських IT компаніях ТОВ «ІНТЕРНЕТ ІНВЕСТ», ТОВ «ПЕРША УКРАЇНСЬКА ЛІЗИНГОВА КОМПАНІЯ», ТОВ «УКРАЇНСЬКІ МАГІСТРАЛЬНІ МЕРЕЖІ» а саме в реєстраторі доменних імен Imena.UA, хостинг-провайдері MiroHost, хостингу доменних зон DNSHosting, українському форумі інтернет-діячів iForum та точці обміну трафіком Giganet, що підтверджено відповідними актами впровадження.

При використанні специфікації *семантичне версіонування баз даних* досягнутий наступний соціально-економічний ефект: була виключена потреба в додатковому інструктажі нових співробітників завдяки усуненню корпоративного маніфесту, який регламентував маркування змін схеми бази даних; відділ технічної підтримки отримав можливість автоматизувати процес впровадження змін у схемах баз даних із використанням CI/CD сервісу; клієнти компанії мають можливість економити час за рахунок автоматичного переходу на нові версії бази даних, так як за номером версії можна визначити чи є вона зворотно сумісною до використовуваної.

Введення у використання категорії *рефакторинг доступу* має наступний соціально-економічний ефект: забезпечено безперебійне обслуговування клієнтів під час впровадження змін, пов'язаних із доступом до даних; мінімізовано час на сервісне обслуговування; забезпечена можливість безперебійно виконувати міграцію даних; регламентовано реакцію на події політики безпеки; створено операції рефакторингу для впровадження змін у правах доступу та масштабування баз даних. За інформацією, наданою однією із компаній, використання запропонованих алгоритмів, під час виділення функцій інформаційної системи в окремий сервіс, скоротило сервісне вікно на термін від 7 хвилин до 258 хвилин завдяки збереженню доступу до даних, які мігрували.

Соціально-економічний ефект від визначення місця рефакторингу в моделях життєвого циклу програмного забезпечення, що відмінні від моделі екстремальне програмування, полягає в наступному: надана можливість зберегти людино-години завдяки визначеному місцю рефакторингу та його цільовому використанню; етап кодування був розділений на окремі підзадачі, завдяки чому спостерігається скорочення часу на виконання робіт. За інформацією, зібраною при приведені апробації, час виконання аналогічних задач скоротився в межах від 7 % до 11 %, крім того розділення етапу кодування на різні підзадачі дає змогу досягти скорочення часу виконання робіт від 6 % до 15 %.

ВИСНОВКИ

Результатом дисертаційного дослідження є вирішення науково-прикладної задачі, що дозволяє підвищити ефективність розробки та супроводу реляційних баз даних для інформаційних систем шляхом розроблення та практичного впровадження нових методів та підходів проведення рефакторингу баз даних.

1. Проаналізовано особливості найчастіше вживаних моделей життєвого циклу програмного забезпечення, а саме каскадна модель, V-модель, інкрементна модель, Rapid Application Development модель, ітеративна модель, спіральна модель. Встановлено, що на даний час розглянуті моделі життєвого циклу програмного забезпечення не виділяли рефакторинг як окремий етап життєвого циклу програмного забезпечення, що призводило до його нераціонального застосування. Тільки в моделі екстремального програмування рефакторинг визначено як окремий прийом.
2. Визначено місце застосування рефакторингу у популярних моделях життєвого циклу програмного забезпечення, що дає змогу уникнути його

нераціонального застосування, в результаті чого зменшено час на його проведення в цілому.

3. Створено специфікацію семантичного версіонування баз даних, що дає можливість економії часу за рахунок автоматизованого переходу на нові версії бази даних.

4. Розроблено нову категорію рефакторингу баз даних – категорію рефакторинг доступу, яке дає змогу впроваджувати зміни, пов’язані з доступом до даних, при адмініструванні системи управління базою даних без втрати працездатності інформаційної системи, скорочує час на сервісне обслуговування та роботи, що пов’язані з безпекою доступу до бази даних.

5. Створено імітаційну модель процесу розробки програмного забезпечення, яка при різних вхідних даних (розміру системи, кількості задач, досвіду розробників) засвідчила, що використання рефакторингу сприяє підвищенню швидкості розробки системи та якості програмного продукту незалежно від кваліфікації команди розробників.

6. Практичне значення результатів роботи підтверджено довідками та актами про їх використання в процесі створення та удосконалення інформаційних систем такими українськими ІТ-компаніями: ТОВ «ІНТЕРНЕТ ІНВЕСТ», ТОВ «ПЕРША УКРАЇНСЬКА ЛІЗИНГОВА КОМПАНІЯ», ТОВ «УКРАЇНСЬКІ МАГІСТРАЛЬНІ МЕРЕЖІ».

Наукові висновки та практичні результати, одержані в ході проведення дослідження відповідно до теми дисертації, пройшли апробацію на міжнародних науково-практичних конференціях і наукових семінарах та викладені у матеріалах наукових конференцій та наукових статтях у виданнях міжнародного значення.

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Статті у наукових фахових виданнях України та закордонних періодичних виданнях, які індексуються у міжнародних наукометричних базах:

1. Струзік В. А. Аналіз засобів забезпечення додаткового захисту корпоративних баз даних / В. А. Струзік, С. В. Грибков, О. В. Харкянен // Вісник Національного університету «Львівська політехніка», серія «Автоматика, вимірювання та керування». – 2017. – №880. – С. 60–67.; ***внесок автора:*** детально описано проблеми захисту корпоративних сховищ та баз даних при їх створенні та супроводу; ***база(и)***: *РИНЦ(eLIBRARY), Index Copernicus International.*

2. Струзік В. А. Дослідження методів і підходів проведення рефакторингу баз даних / В. А. Струзік, С. В. Грибков, А. О. Литвин // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. – 2018. – №30-31. – С. 151–155.; ***внесок автора:*** деталізовано етапи рефакторингу баз даних та описані операції рефакторингу баз даних; ***база(и)***: *РИНЦ(eLIBRARY), Universal Impact Factor, Open Academic Journals Index.*

3. Струзік В. А. Визначення місця рефакторингу в сучасних методологіях розробки інформаційних систем / В. А. Струзік, С. В. Грибков, В. В. Чобану // Вчені записки Таврійського національного університету імені В.І.Вернадського, серія «технічні науки» Том 30 (69). – 2019. – №5. – С. 173–177.; ***внесок автора****: вперше описано місце та роль рефакторингу в моделях життєвого циклу програмного*

забезпечення, що виникли до популяризації рефакторингу; ***база(и)***: *РИНЦ(eLIBRARY), Index Copernicus International.*

4. Струзік В. А. Категорія рефакторинг доступу / В. А. Струзік, С. В. Грибков, В. В. Чобану // Наукові праці НУХТ Том 26. – 2020. – №2 – С. 31–49.; ***внесок автора****: створено та описано нову категорії рефакторингу баз даних, описано основні підходи створення корпоративних інформаційних систем та необхідність проведення рефакторингу в таких системах;* ***база(и)****: Index Copernicus, EBSCOhost, Google Scholar.*

5. Струзік В. А. Специфікація семантичне версіонування бази даних / В. А. Струзік, С. В. Грибков, В. В. Чобану // Вчені записки Таврійського національного університету імені В. І. Вернадського, серія «технічні науки» Том 31 (70). – 2020. – №2. – С. 196–201.; ***внесок автора****: вперше описано специфікацію семантичне версіонування баз даних*; ***база(и)***: *РИНЦ(eLIBRARY), Index Copernicus International.*

6. Struzik V. Evolution of refactoring / V. Struzik, S. Hrybkov, V. Chobanu // Austrian Journal of Technical and Natural Sciences. №7-8 – 2020. – С. 11–16. ***внесок автора****: описано та проаналізовано етапи розвитку рефакторингу, а також його вплив на процес створення програмних продуктів*; ***база(и)***: *РИНЦ(eLIBRARY), CrossRef, CyberLeninka, EBSCOhost, Google Scholar, OpenAIRE, Ulrich's Periodicals Directory, WorldCat.*

Статті та тези доповідей у збірниках праць міжнародних наукових конференцій:

7. Струзік В. А. Рефакторинг баз даних. «Запахи» баз даних. [Електронний ресурс] / В. А. Струзік, Л. Ю. Маноха // Матеріали 82-ї міжнародної наукової конференції молодих учених, аспірантів і студентів «Наукові здобутки молоді — вирішенню проблем харчування людства в XXI столітті», 13–14 квітня 2016 р — К: НУХТ, 2016 р. — 345 с.; ***внесок автора****: проаналізовано «погані запахи бази даних» та необхідність проведення рефакторингу баз даних.*

8. Струзік В. А. Розробка модуля інтелектуального аналізу даних для оцінки мережевої інфраструктури підприємства / В. А. Струзік, С. В. Грибков // Матеріали 4 Міжнародної наукової-практичної конференції «Обчислювальний інтелект (результати, проблеми, перспективи)», 16–18 травня 2017 р., Київ-Черкаси — К. ВПЦ «Київський університет». – 2017. – С. 316.; ***внесок автора****: аналіз роботи корпоративних систем, розробка модуля інтелектуального аналізу даних, який дає можливість прогнозувати навантаження на складові корпоративної системи при реструктуризації та перерозподілі обов'язків окремих структурних підрозділів та підприємства в цілому.*

9. Струзік В. А. Рефакторинг Структури Бази Даних: Методи Та Проблеми / В. А. Струзік // Проблеми Інформатизації Тези Доповідей Восьмої Міжнародної Науково-Технічної Конференції. — 11 — 12 квітня 2017 року. — С. 209.; ***внесок автора****: описано зміни структури бази даних як випадок, коли частіше всього виникає необхідність проведення рефакторингу.*

10. Струзік В. А. Використання версіонування бази даних при проведенні рефакторингу / В. А. Струзік, А. О. Литвин // Матеріали 84 міжнародної наукової

конференції молодих учених, аспірантів і студентів «Наукові здобутки молоді — вирішенню проблем харчування людства в XXI столітті», 23–24 квітня 2018 р. — К.: НУХТ, Ч.2. – 2018. – С. 327.; ***внесок автора****: проаналізовано два основних типу контролю версій та висвітлена необхідність організації контролю версій.*

11. Струзік В. А. Аналіз засобів забезпечення додаткового захисту корпоративних баз даних / В. А. Струзік, С. В. Грибков, О. В. Харкянен // Матеріали VI Міжнародної науково-технічної конференції «Захист інформації й безпека інформаційних систем», 01 — 02 червня 2017 р, Львів — «Львівська політехніка». – 2017. – С. 45–46.; ***внесок автора****: проаналізовано та в загальних рисах описано проблеми захисту корпоративних сховищ та баз даних, разом зі співавторами проведено дослідження програмних продуктів додаткового захисту корпоративних сховищ та баз даних, що представлені на ринку.*

12. Струзік В. А. Дослідження методів і підходів проведення рефакторінгу баз даних / В. А. Струзік, С. В. Грибков, А. О. Литвин // Автоматика — 2017: XXIV Міжнародна конференція з автоматичного управління, м. Київ, Україна, 13–15 вересня 2017 року: тези конференції. Київ: НУБіП. 2017. – 2017. – С. 188–189.; ***внесок автора****: описаний рефакторинг баз даних як один із прийомів створення інформаційних систем та в загальному описано етапи проведення рефакторингу та доцільність його використання.*

13. Струзік В. А. Використання еволюційного підходу при рефакторингу баз даних [Електронний ресурс] / В. А. Струзік, С. В. Грибков, А. О. Литвин // Матеріали IV Міжнародної науково-технічної Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами», 22 листопада 2017 р. — К: НУХТ. – 2017. – 281.; ***внесок автора****: описано еволюційний підхід до побудови баз даних та доцільність використання такого підходу, а також роль та місце рефакторингу в даному підході.*

14. Struzik V. A. Study Of The Methods And Approaches Of Databases Refactoring / В. А. Струзік, С. В. Грибков, А. О. Литвин, В. В. Чобану // Програмовані логічні інтегральні схеми та мікропроцесорна техніка в освіті і виробництві. – 2018. – С. 12–13.; ***внесок автора****: проведено дослідження та короткий опис основних існуючих на сьогодні операцій рефакторингу баз даних.*

15. Струзік В. А. Категорія рефакторинг доступу / В. А. Струзік // Міжнародна науково-технічна конференція студентів, аспірантів та молодих вчених «Комп'ютерні науки, інформаційні технології та системи управління». — 27 — 29 листопада 2019 року — Івано-Франківськ — С. 21-22.; ***внесок автора****: вперше описана нова категорія рефакторингу та передумови її появи.*

АНОТАЦІЯ

Струзік В.А. Вдосконалення технологій проведення рефакторингу баз даних для інформаційних систем. – Рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.06 – інформаційні технології. – Національний університет харчових технологій МОН України, Київ, 2020.

У дисертаційній роботі розглянуті методи та підходи рефакторингу при розробці програмного забезпечення та баз даних.

Досліджено класифікацію сучасних інформаційних систем та розглянуті два підходи побудови сервісів, а саме: монолітний шаблон архітектури та мікросервісний шаблон архітектури. Описані їх переваги та недоліки, а також наведені рекомендації щодо використання різних варіантів взаємодії з базами даних при розробці відповідно до мікросервісного шаблону архітектури. Наведено основні ознаки необхідності проведення рефакторингу, недоліки та проблеми рефакторингу баз даних. Описано процес рефакторингу баз даних.

Науковою новизною є створення нової категорії рефакторингу, яка акумулює в собі операції, які надають можливість контрольовано виконувати зміни у частині контролю доступу при адмініструванні системи управління базою даних, створювати регламентовані процеси реакції на події, що пов’язані з політикою безпеки. Створено специфікацію *семантичне версіонування баз даних* та надано рекомендації щодо її застосування. Визначено місце рефакторингу в популярних моделях життєвого циклу програмного забезпечення, а саме в таких моделях: каскадна модель, V-модель, інкрементна модель, RAD модель, модель екстремальне програмування, ітеративна модель, спіральна модель.

Запропоновані в роботі методи та підходи пройшли апробацію та використовуються під час розробки та супроводу інформаційних систем в українських компаніях, що підтверджено відповідними актами впровадження.

***Ключові слова*:** інформаційна система, рефакторинг, життєвий цикл програмного забезпечення, семантичне версіонування, категорія рефакторинг доступу, операції рефакторингу, бази даних, шаблон архітектури.

ABSTRACT

Vladislav Struzik. Improving database refactoring technologies for information systems. - Manuscript.

Thesis for a Candidate Degree in Engineering on the specialty 05.13.06 - Information Technology. - National University of Food Technologies, Kyiv, 2020.

The methods and approaches of refactoring at development of the software and databases are considered in the dissertation thesis.

The classification of modern information systems is investigated and two approaches of building services are considered, namely: monolithic architecture template and microservice architecture template. Their advantages and disadvantages are described, as well as recommendations on the use of different options for interaction with databases in the development of microservice template architecture. This dissertation is based on the analysis of code refactoring and database refactoring. The main features of the need for refactoring, defects and problems of databases refactoring are given. The process of

refactoring databases is described directly. The necessity of refactoring is substantiated, the recommendations on the choice of the appropriate refactoring operation and on the implementation of refactoring are given. Accordingly, the description and features of the refactoring categories and their operations are provided. The concept of semantic database versioning is formulated and recommendations are given for its application.

The main focus and scientific novelty is the creation of a new category of refactoring, the creation of a specification of semantic versioning of database schema, the determining the place of refactoring in popular software lifecycles.

Access refactorings category is accumulates operations that allow controlled implementation of changes in access control in the administration of the database management system, to create regulated processes of response to events related to security policy without information system downtime.

Semantic versioning of the database schema makes it possible to understand whether the new version of the database is inversely compatible with the old version without further creation and study of corporate manifestos that would regulate the versioning of the database. An important advantage of semantic versioning of the database schema is the possibility of automated implementation of updates.

It have been determined which place of refactoring is the most profitable in terms of time spent on system development and, as a consequence, financial costs. Based on this, the place of refactoring in popular software lifecycle models is defined. The place of refactoring determined in: cascade model, V-model, incremental model, RAD model, iterative model, spiral model.

The approaches and methods proposed in the work have been tested and used during development and support in Ukrainian companies, which is confirmed by the relevant implementing acts.

Keywords: information system, refactoring, software lifecycle, semantic versioning, access refactorings category, refactoring operations, databases, architecture template.