

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»

УДК 004.056

«До захисту допущено»

В.о. завідувача кафедри
_____ М.В.Грайворонський
“ ____ ” _____ 2019 р.

Магістерська дисертація
на здобуття ступеня магістра

зі спеціальності: 125 Кібербезпека

на тему: Методика виявлення та виправлення порушень цілісності схеми бази даних на основі скриптів ініціалізації

Виконав (-ла): студент (-ка) _____ курсу, групи _____
(шифр групи)

Кондратюк Оксана Сергіївна _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник к.т.н., доц. Коломицев Михайло Володимирович _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2019 року

РЕФЕРАТ

Представлена робота має обсяг 111 сторінок, 57 ілюстрацій, 23 таблиці та 26 літературних посилань.

Об'єктом дослідження є база даних з порушеною структурою.

Предметом дослідження є виявлення та виправлення порушень цілісності схеми бази даних.

Метою даної кваліфікаційної роботи є розробка методики виявлення та виправлення порушень цілісності схеми бази даних на основі скриптів ініціалізації.

Методи дослідження - ознайомлення та опрацювання літератури, що представлено монографічними та журнальними матеріалами, електронними ресурсами, які стосуються досліджуваної теми, аналіз різних атак на базу даних та методи реагування, структурування отриманих результатів, дослідження представлення схеми бази даних та дослідження початкових скриптів бази даних для подальшого використання при розробці методики.

Наукова новизна полягає в тому, що розроблена методика є повним рішенням, що дозволяє не тільки виявляти порушення схеми бази даних, а й пропонувати варіанти вирішення, чого не існує в науковому співтоваристві. Дана методика використовує скрипти ініціалізації схеми бази даних для порівняння з поточною схемою, чого також не описано та не реалізовано.

Результати роботи викладені у третьому розділі, що демонструють практичну роботу нової методики, яка буде як виявляти порушення схеми бази даних так і пропонувати варіанти для їх вирішення.

Оскільки, наукова новизна даної роботи запропонувати методику виявлення та виправлення змін схеми бази даних, тому результати можуть бути використані для реалізації корпоративного продукту або продукту з відкритим початковим кодом, який зможе бути корисним як провідним компаніям в ІТ сфері, так і невеликим компаніям, які використовують бази даних в бізнесі.

Ключові слова: база даних, схема бази даних, sql, скрипт ініціалізації.

ABSTRACT

The presented work has 111 pages, 57 figures, 23 tables and 26 literary references.

The object of the study is a database with a modified structure.

The subject of the study is the detection and correction of database schema violation.

The purpose of this qualification work is to develop a methodology for detecting and correcting database schema integrity violations based on initialization scripts.

The methods of studying - familiarization and processing of literature, presented by monographic and journal materials, electronic resources related to the topic under study, analysis of various attacks on the database and response methods, structuring the results, researching the presentation of the database schema and researching the initial database scripts for further use methodology development.

The scientific novelty lies in the fact that the developed methodology is a complete solution that allows not only to identify violations of the database schema, but also to offer solutions that do not exist in the scientific community. This technique uses database schema initialization scripts to compare with the current schema, which is also not described and not implemented.

The results of the work are presented in the third chapter, demonstrating the practical work of the new methodology, which will both detect violations of the database schema and offer options for solving them.

Since the scientific novelty of this work is to propose a methodology for detecting and correcting changes to the database schema, therefore, the results can be used to implement a corporate product or an open source product that can be useful for both leading companies in the IT field and small companies that use databases in business.

Keywords: database, database schema, sql, initialization script..

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінівВ.....	8
Вступ.....	9
1 Аналіз існуючих методів для виявлення та виправлення змін бази даних.....	11
1.1 Поняття бази даних	11
1.2 Типи баз даних.....	13
1.3 Визначення SQL та пов'язані з ним терміни.....	17
1.4 Існуючі загрози на бази даних.....	20
1.5 Code injection атаки	23
1.6 SQL injection атаки	26
1.7 Існуючі методи захисту від загроз на бази даних	31
1.8 Існуючі методи виявлення зміни схеми бази даних	37
1.9 Існуючі методи виправлення та відновлення бази даних	39
Висновки до розділу 1	40
2 Алгоритм виявлення та виправлення порушень цілісності схеми бази даних	42
2.1 Постановка задачі	42
2.2 Загальна схема методики	43
2.3 Етап представлення та аналізування початкових скриптів.....	45
2.4 Етап аналізування та представлення поточного стану схеми бази даних.....	48
2.5 Етап порівняння схем	57
2.6 Етап реагування та виправлення.....	59
Висновки до розділу 2	60
3 Методика виявлення та виправлення порушень цілісності схеми бази даних	61
3.1 Розробка методики виявлення та виправлення порушень цілісності схеми бази даних.....	61
3.2 Програмна реалізація	66
3.3 Результати роботи.....	71
Висновки до розділу 3	82

4 Розробка стартап-проекту	83
4.1 Опис ідеї стартап-проекту	83
4.2 Технологічний аудит ідеї проекту	87
4.3 Аналіз ринкових можливостей запуску стартап-проекту	89
4.4 Розроблення ринкової стратегії проекту	100
4.5 Розроблення маркетингової програми стартап-проекту	104
Висновки до розділу 4	107
Висновки	108
Перелік джерел посилання	109

ПЕРЕЛІК УМОВНИК ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД - База даних

PHP - Hypertext Preprocessor

HTML - HyperText Markup Language

CMS - Content Management System

СУБД - Система управління базами даних

SQL - Structured Query Language

SQLIA - Structured Query Language Injection Attack

ООСУБД - Об'єктно-орієнтована система управління базами даних

SPARQL - Protocol and RDF Query Language

QL - Query Language

PL - Programming Language

DoS - Denial of Service

RCE - Remote Code Execution

LDAP - Lightweight Directory Access Protocol

Xpath - XML Path Language

XML - eXtensible Markup Language

SMTP - Simple Mail Transfer Protocol

XSS - Cross-Site Scripting

SOP - Standard Operating Procedure

CSRF - Cross Site Request Forgery

DBCC CHECKDB

DDL - Data Definition Language

DML - Data Manipulation Language

DCL - Data Control Language

TCL - Transaction Control Language

JSON - JavaScript Object Notation

UML - Unified Modeling Language

ВСТУП

Актуальність роботи. Один з найважливіших критеріїв надійності інформаційної системи - безпека баз даних системи. Атаки, спрямовані на неї, в більшості випадків критичні, тому що можуть частково або повністю порушити працездатність системи. Однією з найлегших та найпоширеніших атак є атака по впровадженню програмного коду. SQL-ін'єкція становить значну частину (приблизно 25%) всіх мережових атак. Тому, існує потреба в сучасних методах захисту від цього типу атак, а також, в разі, все ж таки, несанкціонованих змін – методах виявлення та методах реагування на зміни в БД, тому робота присвячена аналізу наслідків атак та методів виявлення та реагування на дані атаки.

Об'єктом дослідження є база даних структура якої змінюється несанкціонованим шляхом.

Предметом дослідження є виявлення та виправлення порушень цілісності схеми бази даних.

Метою роботи є розробка методики виявлення та виправлення порушень цілісності схеми бази даних на основі скриптів ініціалізації.

Завдання роботи:

1. Проаналізувати існуючі типи баз даних, їх основні аспекти, загрози на бази даних, а саме: SQL injection атаки, їх різновиди та наслідки для реляційних баз даних.
2. Ознайомитися з існуючими методами виявлення та виправлення порушень бази даних, проаналізувати їх недоліки
3. Запропонувати методику виявлення та виправлення порушень цілісності схеми бази даних на основі скриптів ініціалізації.
4. Реалізувати запропоновану методику, підготувати результати роботи даної методики та зробити висновки.

Методи дослідження - ознайомлення та опрацювання літератури, що представлено монографічними та журнальними матеріалами, електронними

ресурсами, які стосуються досліджуваної теми, аналіз різних атак на базу даних та методи реагування, структурування отриманих результатів, дослідження представлення схеми бази даних та створення початкових скриптів бази даних для подальшого використання при розробці методики.

Наукова новизна полягає в тому, що розроблена методика є повним рішенням, що дозволяє не тільки виявляти порушення схеми бази даних, а й пропонувати варіанти вирішення, чого не існує в науковому співтоваристві. Дана методика використовує скрипти ініціалізації схеми бази даних для порівняння з поточною схемою, чого також не описано та не реалізовано.

Результати роботи викладені у третьому розділі, що демонструють практичну роботу нової методики, яка буде як виявляти порушення схеми бази даних так і пропонувати варіанти для їх вирішення.

Практичне значення результатів роботи може бути використане для реалізації корпоративного продукту або продукту з відкритим початковим кодом, який зможе бути корисним як провідним компаніям в ІТ сфері, так і невеликим компаніям, які використовують бази даних в бізнесі.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ДЛЯ ВИЯВЛЕННЯ ТА ВИПРАВЛЕННЯ ЗМІН БАЗИ ДАНИХ

1.1 Поняття бази даних

База даних (БД) - це організована структура, призначена для зберігання, зміни і обробки взаємозалежної інформації, переважно великих обсягів. Бази даних активно використовуються для динамічних сайтів зі значними обсягами даних - часто це інтернет-магазини, портали, корпоративні сайти. Такі сайти зазвичай розроблені за допомогою серверної мови програмування (як приклад, PHP) або на основі CMS (як приклад, WordPress), і не мають готових сторінок з даними за аналогією з HTML-сайтами. Сторінки динамічних сайтів формуються «на льоту» в результаті взаємодії скриптів і баз даних після відповідного запиту клієнта до веб-сервера.[1]

У визначеннях найбільш часто (явно або неявно) присутні такі відмінні ознаки:

1. БД зберігається і обробляється в обчислювальній системі. Таким чином, будь-які некомп'ютерні сховища інформації (архіви, бібліотеки, картотеки і т. П.) базами даних не є.
2. Дані в БД логічно структуровані (систематизовані) з метою забезпечення можливості їх ефективного пошуку і обробки в обчислювальній системі.

Структурованість передбачає явне виділення складових частин (елементів), зв'язків між ними, а також типізацію елементів і зв'язків, при якій з типом елемента (зв'язку) співвідноситься певна семантика і допустимі операції .

3. БД включає схему, або метадані, що описують логічну структуру БД в формальному вигляді (відповідно до деякої метамоделі).[2]

Відповідно до ГОСТ Р ІСО МЕК ТО 10032-2007, «постійні дані в середовищі бази даних включають в себе схему і базу даних. Схема включає в себе опис змісту, структури і обмежень цілісності, що використовуються для

створення і підтримки бази даних. База даних включає в себе набір постійних даних, певних за допомогою схеми. Система управління даними використовує визначення даних в схемі для забезпечення доступу і управління доступом до даних в базі даних »[3]

В контексті баз даних варто розглянути поняття СУБД. Система управління базами даних (СУБД) - це комплекс програмних засобів, необхідних для створення структури нової бази, її наповнення, редагування вмісту і відображення інформації. Найбільш поширеними СУБД є MySQL, PostgreSQL, Oracle, Microsoft SQL Server. СУБД служить інтерфейсом між базою даних та її кінцевими користувачами або програмами, дозволяючи користувачам отримувати, оновлювати та керувати способом організації та оптимізації інформації. СУБД також полегшує контроль баз даних, що дозволяє здійснювати різні адміністративні операції, такі як моніторинг продуктивності, налаштування, резервне копіювання та відновлення.

У реляційній базі даних цифрова інформація про конкретного клієнта впорядковується у рядки, стовпці та таблиці, які індексуються, щоб полегшити пошук відповідної інформації за допомогою SQL запитів. На відміну від цього, графічна база даних використовує вузли та ребра для визначення зв'язків між записами даних і запитами, потребує спеціального синтаксису семантичного пошуку. Станом на сьогодні, SPARQL - єдина семантична мова запитів, яка затверджена Всесвітнім консорціумом веб-сторінок (W3C).

Зазвичай менеджер баз даних надає користувачам можливість контролювати доступ для читання/запису, задавати генерацію звітів та аналізувати використання. Деякі бази даних пропонують відповідність ACID (атомарність, послідовність, ізоляція та довговічність), щоб гарантувати відповідність даних та завершення транзакцій.

1.2 Типи баз даних

Бази даних еволюціонували з часу їх створення в 1960-х роках, починаючи з ієрархічних та мережових баз даних, до 1980-х років з об'єктно-орієнтованими базами даних, а сьогодні - з базами даних SQL і NoSQL та хмарними базами даних.

З одного погляду, бази даних можна класифікувати за типом вмісту: бібліографічний, повний текст, числовий та зображення. При обчислювальній роботі бази даних іноді класифікуються відповідно до їх організаційного підходу. Існує багато різних типів баз даних, починаючи від найбільш поширеного підходу, реляційної бази даних, до розподіленої бази даних, хмарної бази даних, графічної бази даних або бази даних NoSQL.

Розглянемо наступні типи баз даних:[4]

- Реляційна база даних

Реляційна база даних, винайдена Е. Ф. Коддом в IBM в 1970 р., - це таблична база даних, в якій дані визначаються таким чином, щоб вони могли бути реорганізовані та доступні різними способами.

Реляційні бази даних складаються з набору таблиць з даними, що вписуються у заздалегідь задану категорію. Кожна таблиця містить щонайменше одну категорію даних у стовпці, і кожен рядок має певний екземпляр даних для категорій, визначених у стовпцях. Реляційні бази даних легко розширити, і нову категорію даних можна додати після створення оригінальної бази даних, не вимагаючи зміни всіх існуючих програм. Реляційна база даних має як свої переваги, так і недоліки.

- Одним з важливих переваг реляційного підходу є його простота і доступність для розуміння кінцевим користувачем. Єдиною інформаційною конструкцією є таблиця.
- Ще однією важливою перевагою та особливістю є те, що реляційна база даних має строгую статичну типізацію, тобто, наприклад, після

ключових слів `create table` завжди йде назва таблиці, що буде створена, або ж після назви таблиці в дужках ідуть ім'я колонки та її тип даних, що перераховуються через кому. Дана особливість є ключовою для подальшого процесу з даним типом бд. Інші ж бази даних не притримуються даного строгого формату.

- При проектуванні реляційних баз даних застосовуються суворі правила, що базуються на математичному апараті.
- Реляційна модель забезпечує повну незалежність даних. При зміні структури реляційної бази даних зміни, які потрібно зробити в прикладних програмах, як правило, мінімальні.
- Розподілена база даних

Розподілена база даних - це база даних, в якій частини бази даних зберігаються в декількох фізичних місцях, і в якій обробка розповсюджується або реплікується між різними точками мережі.

Розподілені бази даних можуть бути однорідними або неоднорідними. Всі фізичні розташування в однорідній системі розподілених баз даних мають одне і те ж саме апаратне забезпечення і працюють однакові операційні системи та програми баз даних. Апаратне забезпечення, операційні системи або програми бази даних в неоднорідній розподіленій базі даних можуть бути різними в кожному з розташувань.

Також розподілені бази даних можуть бути фрагментовані або тиражовані. Фрагментована, або секціонована (англ. Partitioned database) – це база даних, в якій методом розподілу даних є фрагментованість (секціонування), вертикальне чи горизонтальне. Тиражована (англ. Replicated database) – це база даних, в якій методом розподілу даних є тиражування (реплікація).

- Хмарна база даних

Хмарна база даних - це база даних, оптимізована або побудована для віртуалізованого середовища, або в гібридній хмарі, у відкритій або в приватній хмарі. Хмарні бази даних надають такі переваги, як можливість платити за

обсяг сховища і пропускну здатність для кожного використання, а також забезпечують масштабованість за потребою, а також високу доступність.

Хмарна база даних також надає підприємствам можливість підтримувати бізнес-застосунки в розгортанні програмного забезпечення.

- База даних NoSQL

Бази даних NoSQL корисні для великих наборів розподілених даних. Вони ефективні для проблем з великими показниками продуктивності даних, які не розроблені для реляційних баз даних. Вони найбільш ефективні, коли організація повинна проаналізувати великі шматки неструктурованих даних або даних, які зберігаються на декількох віртуальних серверах у хмарі. NoSQL - це нова категорія систем управління базами даних. Його основною характеристикою є недотримання концепцій реляційних баз даних. NOSQL означає «не тільки SQL». Концепція баз даних NoSQL виросла з інтернет-гігантами, такими як Google, Facebook, Amazon та т. д., які мають справу з гігантськими обсягами даних. Коли використовується реляційна база даних для величезних обсягів даних, система починає сповільнюватися з точки зору часу відгуку. Щоб подолати це, ми, звичайно, могли б «розширити» наші системи, модернізуючи наше існуюче обладнання. Альтернативою вищевказаної проблеми був би розподіл навантаження на нашу базу даних на кілька хостів по мірі збільшення навантаження. Це відомо як «масштабування». Бази даних NOSQL - це нереляційні бази даних, які масштабуються краще, ніж реляційні бази даних, і розробляються з урахуванням веб-додатків. Вони не використовують SQL для запиту даних і не слідують строгим схемам, таким як реляційні моделі. З NoSQL функції ACID (атомарність, узгодженість, ізоляція, довговічність) не завжди можуть бути гарантовані.

- Об'єктно-орієнтована база даних

Об'єктно-орієнтована СУБД - цей тип підтримує зберігання нових типів даних. Дані, що підлягають збереженню, мають форму об'єктів. Об'єкти, що зберігаються в базі даних, мають атрибути (тобто стать, вік) та методи, що

визначають, що робити з даними. PostgreSQL - приклад об'єктно-орієнтованої реляційної СУБД.

Переваги використання ООСУБД:

- Відсутня проблема невідповідності моделі даних в веб-застосунку і БД (impedance mismatch). Всі дані зберігаються в БД в тому ж вигляді, що і в моделі застосунку.
- Не потрібно окремо підтримувати модель даних на стороні СУБД.
- Всі об'єкти на рівні джерела даних строго типізовані. Більше ніяких строкових імен колонок! Рефакторинг об'єктно-орієнтованої бази даних і працюючого з нею коду тепер автоматизований, а не одноманітний і нудний процес.
- Графо-орієнтована база даних

Графо-орієнтована база даних, або графова база даних, є типом бази даних NoSQL, яка використовує теорію графів для зберігання, відображення і запиту взаємозв'язків. Графо-орієнтована база даних - це, в основному, набори вузлів і ребер, де кожен вузол являє сутність, а кожне ребро являє зв'язок між вузлами.

Графо-орієнтовані бази даних стають все більш популярними для аналізу взаємозв'язків. Наприклад, компанії можуть використовувати графічну базу даних для збору даних про клієнтів з соціальних мереж.

Графо-орієнтовані бази даних часто використовують SPARQL, декларативну мову програмування і протокол для аналізу графових баз даних. SPARQL має можливість виконувати всю аналітику, яку може виконувати SQL, а також може використовуватися для семантичного аналізу, вивчення відносин. Це робить його корисним для виконання аналітики наборів даних, які мають як структуровані, так і неструктуровані дані. SPARQL дозволяє користувачам виконувати аналітику інформації, що зберігається в реляційній базі даних, а також відносин один-одного (FOAF), PageRank і найкоротшого шляху.

1.3 Визначення SQL та пов'язані з ним терміни

SQL (мова структурованих запитів) - це мова високого рівня, основа якої сильно залежить від реляційної алгебри і реляційного числення. SQL складається з декларативних елементів, таких як запити, вирази, пропозиції, оператори і т.д. SQL широко відомий як потужна мова запитів. Основна відмінність між мовою запитів (QL) і мовою програмування (PL) полягає в тому, що QL їх не можна використовувати для складних обчислень, і вони мають дуже хорошу ефективність для обробки великих наборів даних.[5]

Реалізація мови запитів заснована на реляційній алгебрі (операційна частина) і реляційному численні (декларативна частина). Реляційна алгебра відноситься до специфікації послідовності операцій для виконання певного запиту, тоді як реляційне числення відноситься до специфікації необхідного висновку без будь-якої інформації про послідовність операцій, необхідної для обробки запиту [6].

Подібно до інших алгебр, деякі оператори є примітивними, а інші, будучи визначені через примітивні, є похідними від них. В реляційній алгебрі Кодда визначено такі шість примітивних операторів: вибірка, проекція, декартів добуток, об'єднання та різниця і перейменування (насправді, Кодд відмовився від включення оператора перейменування, однак, розробники ISBL навели приклади необхідності його включення). Шість операторів є фундаментальними в тому сенсі, що жоден із них не можна відкинути без втрати потужності. Багато інших операторів було визначено комбінацією цих шести. Серед найважливіших можна назвати: перетин множин, ділення та природне об'єднання. Насправді, ISBL дала підстави для заміни декартового добутку природнім об'єднанням, окремим випадком якого є декартів добуток [7].

У таблиці 1.1 наведено кілька прикладів примітивних операторів, які використовуються при розробці оператора запиту.

Таблиця 1.1 - Приклад примітивних операторів[7]

Вибірка	Вибирає підмножину рядків в таблиці.
Проекція	Видаляє атрибути з таблиць, яких немає в списку. При виконанні проекції виділяється "вертикальна" вирізка відносини-операнда з природним знищенням потенційно виникаючих кортежів-дублікатів.
Декартів добуток	Дозволяє поєднання реляційного пошуку
Встановлення різниці	Визначає взаємовиключні зв'язки
Об'єднання	Відношення з тим же заголовком, що і у сумісних по типу відносин A і B, і тілом, що складається з кортежів, що належать або A, або B, або обом відносинам. Синтаксис: A UNION B

Додаткові оператори, такі як перетин, поєднання, ділення і перейменування, також корисні.

Підводячи підсумок, можна сказати, що це реляційна модель, яка строго визначена простотою і здатністю операційної алгебри ефективно виконувати всі завдання, пов'язані з базою даних, з найкращою можливою оптимізацією. В результаті, SQL є мовою спілкування для доступу до систем баз даних.

Після розгляду визначень SQL і пов'язаних з ними термінів ми розглянемо базовий метод обробки в мережі і обговоримо архітектуру, необхідну для обробки SQL. Ми опишемо обробку веб-застосунку в наступній послідовності:

1. Після того як клієнт (звичайний веб-користувач, підключений до Інтернету) вводить веб-адресу в застосунку веб-браузера, сервер

(комп'ютер, на якому зберігаються веб-сторінки, сайти або додатки) відправляє копію форми клієнту.

2. Користувачі веб-браузера вводять дані в цю форму і відправляють їх на сервер.
3. Сервер запускає сценарій для форми і, коли він визначає, що це доречно, надає клієнтові доступ до основного застосунку або до бази даних.

Одним із прикладів архітектури, необхідної для обробки веб-застосунку, є архітектура, що показана на рисунку 1.1:

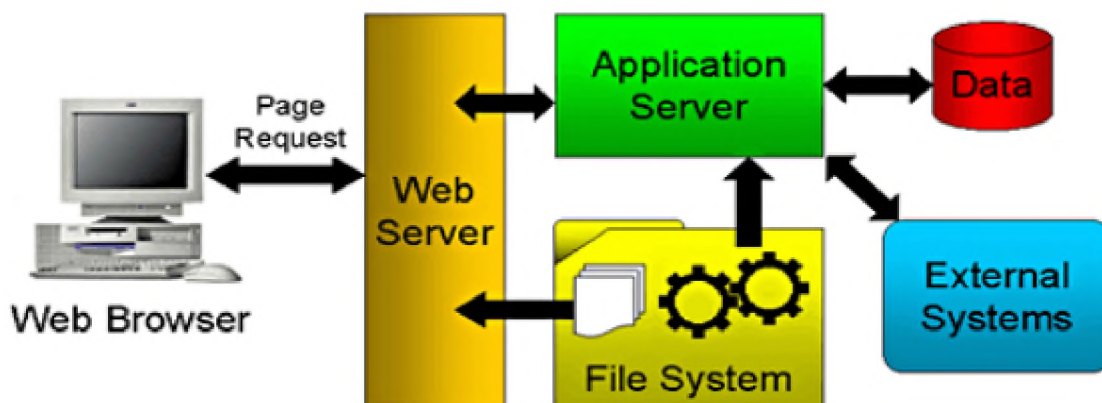


Рисунок 1.1 - Канонічна веб-архітектура[8]

Дана веб-архітектура в цілому поділяється на три рівні обробки. Веб-браузер на рисунку 1 представляє рівень представлення(показу, презентації). Веб-сервер, сервер застосунків і файлова система складають рівень сервера. Нарешті, дані представляють рівень бази даних системи веб-застосунків. Функції кожного з цих рівнів описані в [9] і узагальнені нижче.

1. Рівень презентації: містить діалог з користувачами системи. Він має справу з користувацьким введенням, відповіддю від системи і є лицем веб-служби для користувача.

2. Рівень сервера: це основа всієї системи баз даних, оскільки вона містить логіку, необхідну для роботи бази даних.
3. Рівень бази даних: це точка зберігання, де зберігаються всі дані.

Заходи безпеки можуть бути реалізовані на рівні презентації (застосунків), на рівні сервера (простір сервера) або на рівні бази даних (фізичне розташування бази даних).

Захисні міри, які можуть застосовуватися до веб-застосунків, можуть включати виправлення програмного забезпечення для процесів збору даних на рівні презентації, шифрування даних на рівні бази даних. Однак у багатьох випадках заходи безпеки на рівні сервера системи веб-додатків особливо підходять для запобігання різних типів атак.

1.4 Існуючі загрози на бази даних

На даний момент існує велика кількість атак на бази даних. Вони проводяться з метою отримання важливої інформації, для досягнення результату можуть бути використані адміністратор бази даних. За даними OWASP, основні 10 з них описані в [10] та наведені нижче:

1. Помилки конфігурації хмарної бази даних

Помилки конфігурації хмарної бази даних викликані небезпечно налаштованими хмарними базами даних або службами зберігання. IP-адреси служби Public Cloud не є секретними і постійно перевіряються на наявність вразливостей зловмисниками і дослідниками безпеки. Багато порушень пов'язані зі сховищами даних, про які організації не знали, чи які були створені небезпечно на неконтрольованій основі.

2. SQL-ін'єкція

Уразливості SQL-ін'єкцій виникають, коли код додатка містить динамічні запити до бази даних, які безпосередньо включають вводимі користувачем дані. Це руйнівна форма атаки, і тестери BSI Penetration регулярно знаходять

вразливі додатки, які дозволяють повністю обійти аутентифікацію і витягти всю базу даних.

3. Слабка аутентифікація

Слабка аутентифікація має багато аспектів, від перебору паролів до небезпечного зберігання облікових даних бази даних, використовуваних застосунком. Дана загроза може існувати через потребу часто міняти паролі, що може призвести до того, що користувач буде використовувати паролі, що легко запам'ятовуються і відповідно передбачувані паролі. Причиною може бути відсутність багатфакторної аутентифікації та зберігання паролів у відкритому вигляді.

4. Зловживання привілеями

Користувачі можуть зловживати законними правами доступу до даних в несанкціонованих цілях. Наприклад, користувач в продажах з правами на перегляд окремих записів клієнтів може використовувати цей привілей для вилучення всіх записів клієнтів для передачі конкуренту. Належна політика найму зменшить ймовірність цього, але вона повинна забезпечуватися технічними заходами і ефективною реєстрацією і моніторингом для виявлення зловживань.

5. Надмірні привілеї

Якщо користувачі мають привілеї, які перевищують вимоги їх посадових функцій, ці привілеї можуть бути порушені окремою особою або зловмисником, який скомпрометує їх обліковий запис. Коли люди переміщують ролі, їм можуть бути надані нові привілеї, в яких вони потребують, а ті, які їм більше не потрібні, будуть видалені.

6. Неповноцінна реєстрація і слабкий аудит

Реєстрація та аудит є ключовими для запобігання та виявлення неправомірного використання і забезпечення адекватного розслідування підозрюваних компрометації даних. У цьому контексті ведення журналу - це збір даних, а аудит - це той, хто насправді дивиться на це. Необхідно думати, як ваші дані журналу будуть захищені. Якщо все це знаходиться в базі даних

програми і є скомпрометованим, зловмисник може стерти або підробити дані журналу. Журнали можуть містити конфіденційну інформацію, що є небезпечним.

7. Відмова в обслуговуванні

DoS (відмова в обслуговуванні) - хакерська атака на обчислювальну систему з метою довести її до відмови, тобто створення таких умов, при яких сумлінні користувачі системи не зможуть отримати доступ до надаваних системних ресурсів (серверів) , або цей доступ буде утруднений. DoS викликається просто: або база заповнюється «сміттєвими» записами, або, що набагато небезпечніше, вона просто видаляється. Другий випадок особливо цікавий, якщо з яких-небудь причин не робилися (або не перевірялися!) бекапи.

8. Неправильне управління виправленнями

Дефекти виникають у всіх типах програмного забезпечення, і операційні системи, такі як Windows, а також системи управління базами даних, такі як SQL Server, не є винятком. Якщо говорити про операційну систему і систему управління базами даних, можна сказати, що вони містять помилки, і ці помилки можуть бути використані людьми зі зловмисними намірами. Якщо зловмисник знайде помилку, яка викликає уразливість, то він зможе отримати доступ до вашого сервера. І як тільки ця вразливість публікується, що часто відбувається в той момент, коли виробник програмного забезпечення робить виправлення доступним, його можуть використовувати всі. Гірше того, незабаром після публікації уразливості звичайні інструменти автоматизації зловмисників починають включати в себе експлойти для її усунення. Як тільки це відбудеться, все, навіть люди з невеликим досвідом хакерства або без нього, зможуть використовувати цю уразливість проти вас.

9. Remote Code Execution (RCE)

Є, мабуть, найнебезпечнішим вектором атаки, на щастя, нечасто зустрічається. Зазвичай виконується з метою отримання shell'a і, отже, контролю над сервером цілком. Часто RCE здійснюється вже після атаки Privilege Escalation і через слабкі налаштування прав доступу в системі, але це

відбувається не завжди так. Для реалізації атаки зловмисник завантажує файл-шкідник і або запускає його віддалено, або сам одним з доступних чином «чіпляється» до нього.

Наслідки можуть бути різними. Хтось починає використовувати сервер для майнінга криптовалюта, хтось може налаштувати реплікацію БД на «свій» сервер, а хтось просто піде далі і постараться отримати управління іншими серверами в локальній мережі.

10. Небезпечне резервне копіювання

Крадіжка стрічок з резервними копіями баз даних і жорстких дисків вже давно викликає занепокоєння, але виникли нові загрози доступності даних, і їх не можна ігнорувати. Всі резервні копії повинні бути зашифровані для захисту конфіденційності та цілісності даних, і це повинно включати належне управління ключами. Ключі не повинні потрапляти в чужі руки, але повинні бути доступні при необхідності для відновлення даних. Якщо ж говорити про стійкість в хмарних сервісах, наприклад, гео-реплікація, не те ж саме, що резервне копіювання. Зловмисник може видалити стільки хмарної інфраструктури і даних про клієнтів, що організація не зможе вижити.

1.5 Code injection атаки

Впровадження коду - це загальний термін для типів атак, які складаються з введення коду, який потім інтерпретується / виконується застосунком. Цей тип атаки використовує погану обробку ненадійних даних і стає можливими через відсутність належної перевірки даних введення / виводу, наприклад:

- дозволені символи (стандартні класи регулярних виразів або призначені для користувача);
- формат даних;
- кількість очікуваних даних.

Включення коду може бути використане зломисником для введення(включення) коду в комп'ютерну програму, щоб змінити хід її виконання. Наприклад, включення коду використовується для поширення комп'ютерних хробаків.

Включення коду трапляється тоді, коли програма надсилає неперевірені дані інтерпретатору. Недоліки включення коду дуже поширені в унаслідкованому коді. Вони часто трапляються у SQL, LDAP, Xpath, або NoSQL запитах; командах операційної системи; синтаксичних аналізаторах XML, заголовках SMTP, аргументах програми. Включення коду легко виявити при перегляді коду, проте його дуже важко виявити тестуванням. Сканери та фузери допомагають зломисникам виявляти вразливості включення коду

Включення коду може призвести до пошкодження чи втрати даних, відсутності звітності або відмови в доступі. Інколи включення коду може призвести навіть до зміни хосту.

Деякі типи включення коду призводять до помилок інтерпретації, надаючи спеціальне значення простому вводу користувача. Це чимось схоже на нездатність розрізняти імена і звичайні слова. За тим же принципом в деяких видах вставленого коду важко розрізнити ввід користувача і системні команди.

Техніка включення коду є поширеною при зломі з метою отримання інформації, отриманні привілейованого або анонімного доступу до системи.

Включення коду можна використовувати у зловмисних цілях, зокрема:

- Довільно змінювати вміст бази даних через так звані SQL інєкції. Наслідком може бути як порушення роботи сайту так і компрометація конфіденційних даних.
- Встановлення шкідливих програм або виконання шкідливого коду на сервері через включення скрипт коду сервера(наприклад PHP чи ASP).
- Отримання доступу до кореневої папки, використовуючи вразливості включення Shell.
- Атаки інтернет-користувачів за допомогою включення HTML/Script(міжсайтовий скриптинг).

Користувачі можуть і не знати, що вони роблять включення коду, бо їхній ввід не був врахований розробниками системи. Наприклад:

- Коректні вхідні дані(на думку користувача) можуть містити марковні символи, або слова, що були зарезервовані програмістом для певних значень (це може бути символ "&" в назві компанії або символ лапок).
- Користувач може надіслати файл невірного формату як вхідні дані. І хоч цей файл працює коректно, він заразить системі, яка отримує файл.[11]

1.5.1 Типи Code injection атак

- Міжсайтовий скриптинг

Міжсайтовий скриптинг (XSS) - це атака з використанням коду, коли шкідливий код впроваджується на веб-сайт і виконується в браузері. Зловмисник вставляє скрипт в браузер жертви, і при доступі до веб-сайту скрипт потім виконується на комп'ютері жертви. Через XSS зловмисник може дистанційно керувати браузером атакованого об'єкта. Це спосіб обійти концепцію стандартної робочої процедури (SOP). Наприклад, всякий раз, коли HTML-код генерується динамічно, а користувацький ввід не очищається і відображається на сторінці, зловмисник може вставити свій HTML-код на цю сторінку [12].

- Підробка міжсайтових запитів

Це атака, при якій зловмисник обманює користувача, виконуючи дії, які корисні зловмисникові при доступі до веб-застосунку. Уразливість CSRF дозволяє зловмисникові змусити користувача виконувати дії, які зловмисник хоче виконати, коли користувач заходить на веб-сайт. Наприклад, злочинець може підробити запит на переказ коштів на веб-сайт.

- Shell ін'єкції

Shell ін'єкції названі так завдяки командній оболонці Linux, але це стосується всіх операційних систем, які дозволяють запуск програм з

командного рядка. Типові функції, пов'язані з shell ін'єкціями: `system()`, `StartProcess()`, і `System.Diagnostics.Process.Start()`.

- SQL Injection

SQL ін'єкція - це атака, що є різновидом атаки з впровадженням коду. Цей експлойт виконується шляхом додавання коду SQL у вхідні дані користувача для отримання доступу до неавторизованих ресурсів. SQLIA можуть виникати, коли запит створюється шляхом об'єднання введених користувачем даних, таких як дані, введені в веб-форму, з ненавмисними даними, включаючи дані URL (Uniform Resource Locator), дані, отримані з файлів cookie і т. д., без належної перевірки. Учасники Rain Forest Puppy, black-hat спільноти, були першими, хто коли-небудь опублікував інформацію про SQLIA в своїй статті «Уразливості NT Web Technology» [13]. SQL-ін'єкція є однією з улюблених атак для багатьох кіберзлочинців, тому що вона може бути виконана віддалено. Комерційно доступні інструменти виявлення вразливостей також доступні зловмисникам, і за допомогою цих ресурсів зловмисник може знайти лазівки в системі безпеки і веб-уразливості за долі секунди. SQL є дуже гнучкою мовою, і ці атаки можуть бути надзвичайно скритними і можуть проходити через брандмауери і системи запобігання вторгнень без особливих зусиль [14].

1.6 SQL injection атаки

Відносна поширеність деяких з цих атак показана на малюнку нижче. Ці дані були отримані від HACKING & TRICKS, блогу про взлом і комп'ютерну безпеку by Nirav Desai, в записі від 8 січня 2013 року [2].

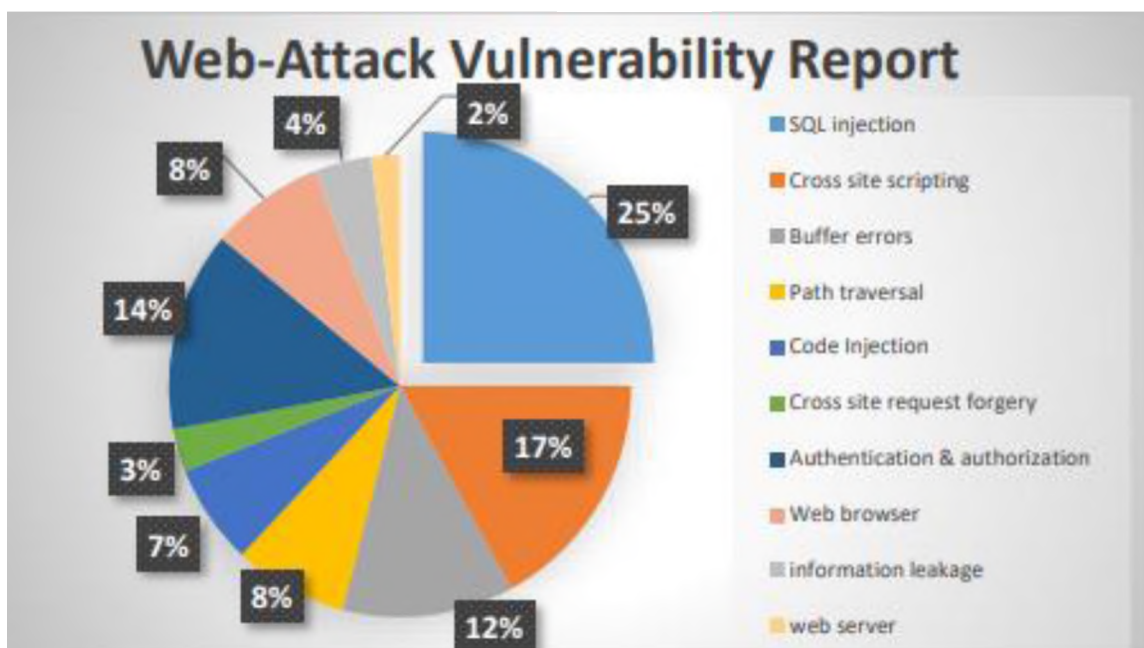


Рисунок 1.2 - Звіт про вразливості веб-атак[2]

Очевидно, що SQL-ін'єкція становить значну частину (приблизно 25%) всіх мережових атак. Тому далі детальніше розглянемо саме SQL injection атаки.

Як вже зазначалось вище, SQL injection є підмножиною code injection атаки і використовує переваги синтаксису SQL для включення команд, які можуть читати чи змінювати базу даних або змінити значення оригінального запиту. Впровадження SQL, залежно від типу СУБД та умов впровадження, може дати можливість атакуючому виконати довільний запит до бази даних (наприклад, прочитати вміст будь-яких таблиць, видалити, змінити або додати дані), отримати можливість читання та/або запису локальних файлів та виконання довільних команд на сервері.

Атака типу впровадження SQL може бути можлива за некоректної обробки вхідних даних, що використовуються в SQL-запитах. Розробник застосунків, що працюють з базами даних, повинен знати про таку вразливість і вживати заходів протидії впровадженню SQL.

Для прикладу, розглянемо веб-сторінку, що має два поля для введення імені користувача і пароля. Насправді код сторінки згенерує SQL запит, щоб перевірити існує такий користувач і чи належний пароль він ввів:

```
SELECT UserList.Username  
FROM UserList  
WHERE UserList.Username = 'Username'  
AND UserList.Password = 'Password'
```

Якщо запит повертає рядки, то доступ надається. Однак, якщо зловмисник введе валідне ім'я користувача і валідний код ("password" OR '1'='1') в поле "Password", тоді запит буде мати вигляд:

```
SELECT UserList.Username  
FROM UserList  
WHERE UserList.Username = 'Username'  
AND UserList.Password = 'password' OR '1'='1'
```

В цьому прикладі, припускається, що поле "Password" є пусте або містить нешкідливий рядок символів. Вираз '1'='1' завжди буде істинним, тому багато рядків повернуться, тим самим надаючи доступ.

Ця техніка може бути удосконалена, дозволяючи наприклад записувати декілька виразів або навіть завантажувати і запускати зовнішні програми.[15]

1.6.1 Типи SQL injection атак

Типи SQL injection атак описані в [16] і наведені нижче:

- Класична SQL Injection - проста і легка в експлуатації. Дозволяє зловмисникові атакувати БД і відразу бачити результат атаки. Останнім часом зустрічається нечасто. Можливість проведення класичної SQL-ін'єкції багато в

чому спрощує отримання корисної інформації. Проведення атаки з використанням класичної техніки експлуатації SQL Injection відбувається з використанням оператора union або з використанням поділу SQL запитів (крапка з комою). Але не завжди вразливість типу SQL Injection можливо експлуатувати подібним способом. У таких випадках вдаються до техніки експлуатації уразливості «сліпим» методом.

- Сліпа SQL-ін'єкція (blind SQL Injection) - з'являється в тому випадку, коли вразливий запит є деякою логікою роботи програми, але не дозволяє вивести будь-які дані в повертаєму сторінку Web застосунком. Сліпу SQL-ін'єкцію за своїми можливостями можна порівняти з класичною технікою впровадження операторів SQL. Аналогічно класичній техніці експлуатації подібних вразливостей, blind SQL Injection дозволяє записувати і читати файли, отримувати дані з таблиці, але тільки читання в даному випадку здійснюється посимвольно. Класична техніка експлуатації подібних вразливостей ґрунтується на використанні логічних виразів true / false. Якщо вираз істинний, то Web застосунок поверне один вміст, а якщо вираз є хибним, то інший. Покладаючись на відмінності виведення при істинних і хибних конструкціях в запиті, стає можливим здійснювати посимвольний перебір будь-яких даних в таблиці або в файлі.

Уразливість blind SQL Injection з'являється в наступних випадках:

- атакуючий не може контролювати дані, що виводяться користувачеві в результаті виконання уразливого SQL-запиту;
- коли ін'єкція потрапляє в два різних SELECT-запити, які в свою чергу здійснюють вибірку з таблиць з різною кількістю стовпців;
- коли використовується фільтрація склеювання запитів.

- Error-based SQL Injection - трохи складніший і витратніший за часом тип атаки, що дозволяє, на основі виведених помилок СУБД, отримати інформацію про всю БД і дані, що зберігаються в ній. Експлуатується, якщо хтось в поспіху забув відключити вивід помилок. Суть Error-based полягає в тому, що ми можемо витягувати потрібну нам інформацію з запиту за допомогою перегляду

помилки роботи викликаємих функцій. Однією з таких функцій в БД MySQL є `extractvalue ()`. Error-based SQL Injection - це найшвидша техніка експлуатації сліпих SQL ін'єкцій. Суть даної техніки полягає в тому, що різні СУБД при певних некоректних SQL-виразах можуть поміщати в повідомлення про помилку різні запитовані дані (наприклад, версію бази даних). Дана техніка може використовуватися в разі, коли будь-яка помилка обробки SQL-виразів, здійснювана в СУБД, повертається назад вразливим застосунком.

- Boolean-based SQL Injection - одна з «сліпих» ін'єкцій. Суть атаки зводиться до додавання спеціального підзапиту в вразливий параметр, на який БД буде відповідати або True, або, несподівано, False. Атака не дозволяє відразу вивести всі дані БД «на екран» зломисникові, але дозволяє, перебираючи параметри раз по раз, отримати вміст БД, хоча для цього буде потрібно часовий відрізок співставимий з вмістом БД.
- Time-based SQL Injection - наступна з «сліпих» ін'єкцій. В даному випадку зломисник додає підзапит, що приводить до уповільнення або паузі роботи БД при деяких умовах. Таким чином, атакуючий, порівнюючи час відповіді на «True» і на «False» запити, символ за символом може отримати весь вміст БД, але часу піде на це більше, ніж в разі експлуатації Boolean-based атаки.
- Out-of-band SQL Injection - рідкісний тип. Атака може бути успішна тільки при певних обставинах, наприклад, якщо сервер БД може генерувати DNS- або HTTP-запити, що зустрічається нечасто. Також, як і Blind SQL, дозволяє посимвольно збирати інформацію про дані, що там зберігаються.

1.6.2 Вектори SQL injection атак

Вектори SQL injection атак описані в [17] і продемонстровані нижче:

1. SQL маніпуляції

У цій атаці маніпулюють виразом, наступним за словом «where», щоб створити поведінку, неочікувану програмістом бази даних (наприклад, складання виразу

where оператором union може забезпечити доступ до даних, до яких у користувача не повинно бути доступу.)

2. Впровадження коду

У цій атаці новий оператор SQL об'єднується з раніше представленим оператором SQL (наприклад, додаючи оператор execute в кінці загального оператора.) Обмеження цього виду SQLIA полягає в тому, що база даних повинна підтримувати кілька операторів SQL за запит.

3. Ін'єкція виклику функції

Це вторинна ін'єкція атаки, при якій зловмисник використовує вбудовані функції бази даних, щоб викликати SQLIA, яка маніпулює даними відповідно до потреб зловмисника.

4. Атака переповнення буфера

У цьому випадку дані, введені в якості вхідних даних, сильно перевищують межі пам'яті планованого простору зберігання. Вона буде перезаписувати покажчики даних і може також використовуватися для вказівки на виконуваний файл, що змушує систему виконувати будь-який файл, який є наміром зловмисника.

1.7 Існуючі методи захисту від загроз на бази даних

Способів захисту від атак багато, але не один з них не дає 100% гарантії захищеності. Адже на кожну дію є протидія, і особливо зацікавлений досвідчений хакер напевно знайде спосіб, як отримати доступ до вашої БД. Проте давайте розглянемо основні з них:[18]

- Створення менш привілейованого користувача

У більшості випадків відвідувачам не потрібно видаляти або оновлювати інформацію. Уявімо інтернет-магазин. Користувач може зробити запит (SELECT) або залишити замовлення (INSERT).

Таким чином, краще створити кілька різних користувачів. Для адміністратора надати всі привілеї, а для звичайного користувача обмежені. Приклад з'єднання для різних користувачів

```

<?php

$visitorDbLink = mysql_connect('host', 'general_user', 'general_user_pass');
$adminDbLink = mysql_connect('host', 'admin_user', 'admin_pass');

?>

```

Рисунок 1.3 – Приклад створення менш привілейованого користувача[18]

Тепер можна використовувати \$ visitorDbLink для регулювання доступу до бази даних для відвідувачів, і використовувати \$ adminDbLink для доступу в якості адміністратора.

- Відключення повідомлень про помилки

Перш за все необхідно уникати вбудованої MySQL функції mysql_error (). Розумний взломщик може вгадати деякі параметри бази даних з повідомлення про помилку, а іноді і побачити параметри з'єднання. Краще використовувати mysql_error () тільки на стадії розробки. Але прибрати її, коли запускаєте сайт на сервері.

Також потрібно відключити звіти про помилки в PHP. Це робиться одним рядком:

```

<?php

// Стякнучить виведє ошибок
error_reporting(0);

?>

```

Рисунок 1.4 – Приклад відключення звіту про помилки[18]

А краще створити власне повідомлення про помилку.

```
<?php

if(!mysql_query($statement))
{
    echo 'Извините, но сервер не доступен!';
}

?>
```

Рисунок 1.5 – Приклад створення власного повідомлення про помилку[18]

В результаті, користувач не дізнається з повідомлення про помилку ніякої важливої інформації, такої як, ім'я бази даних, ім'я таблиці, ім'я користувача та інших. Тим самим ми ускладнюємо хакеру можливість дізнатися структуру SQL-запиту, використовуючи різні ін'єкції.

- Використання параметризованих запитів

Замість того, щоб підставляти значення SQL-заяв напряду, підставляйте їх параметризовані значення, в такий спосіб:

```
<?php

$db_connection = new mysqli("localhost", "user", "pass", "db");
$stmt = $db_connection->prepare("SELECT * FROM customers WHERE id = ?");
$stmt->bind_param("i", $id);
$stmt->execute();

?>
```

Рисунок 1.6 – Приклад використання параметризованих запитів[18]

- "i" - тільки цілі числа (int)
- "d" - числа з плаваючою точкою (double)
- "s" - рядки (string)
- "b" - надсилається в пакетах (blob)

- Використання збережених процедур

Використання збережених процедур, теж може допомогти знизити ризик виникнення атаки. Використання процедур показано на наступному прикладі:

```
<?php

$sqlStatement = "
    CREATE PROCEDURE HUGEORDER
    (
        id INT ,
        quantity INT,
        price DECIMAL(6,2)
    )
    BEGIN
        DECLARE discount_percent DECIMAL(6,2);
        DECLARE discounted_price DECIMAL(6,2);
        SET discount_percent = 10;
        SET discounted_price = price - discount_percent/100*price;
        IF quantity > 500 THEN
            SET discounted_price = discounted_price - 0.25 * q;
        END IF;
        UPDATE fashion_products
        SET product_price = discounted_price WHERE product_id = id;
        Select * from fashion_products;
    END;
";

?>
```

Рисунок 1.7 – Приклад використання збережених процедур[18]

- Використання функцій блокування

Можна використовувати функцію `mysql_real_escape_string ()` для обробки зовнішніх даних. Наприклад:


```
<?php
$username = mysql_real_escape_string($username, $dbLink);
?>
```

Рисунок 1.8 – Приклад використання функцій блокування[18]

Це дуже потужна вбудована функція PHP, здатна запобігти SQL-ін'єкції в більшості випадків. Можна спробувати впровадити SQL-код після використання `mysql_real_escape_string()` і тестувати на уразливості. Ця функція відкидає безліч розумних методів атак, які використовуються зловмисниками.

- Застосування регулярних виразів

Регулярні вирази використовуються для того, щоб привести вхідні дані до одного шаблону. Наприклад, тут ми перевіряємо email клієнта на валідність і відкидаємо можливість для SQL-ін'єкцій.

```
<?php

if(!preg_match("/^[0-9a-z\_\\.\\-]+@[\\-a-z0-9]+\\.([a-z]{2,})$/i", $email))
{
    echo "INVALID Email Address!";
    return;
}

?>
```

Рисунок 1.9 – Приклад застосування регулярних виразів[18]

Також можна користуватися вбудованими функціями PHP `is_array()`, `is_bool()`, `is_double()`, `is_float()`, `is_int()`, `is_integer()` та іншими, для перевірки даних користувача.

- Фільтрація вхідних даних[19]

Нехай маємо запит:

```
statement := 'SELECT * FROM users WHERE id = ' + id + ';;';
```

Рисунок 1.10 – Приклад SQL запиту[19]

В даному випадку поле `id` має числовий тип, і його найчастіше не беруть в лапки. Тому «закавичування» і заміна спецсимволів на ескапе-послідовності не проходить. В такому випадку допомагає перевірка типу; якщо змінна `id` не є числом, запит взагалі не повинен виконуватися.

Для PHP цей метод буде виглядати так:

```
$query = 'SELECT * FROM users WHERE id = ' . (int)$id;
```

Рисунок 1.11 – Приклад фільтрації вхідних даних[19]

- Екранування спеціальних символів

Екранування символів - заміна в тексті керуючих символів на відповідні текстові підстановки. Припустимо, що код, генеруючий запит (на мові програмування Паскаль), виглядає так:

```
statement := 'SELECT * FROM users WHERE name = "' + userName + '";;';
```

Рисунок 1.12 – Приклад генеруючого запиту[19]

Щоб впровадження коду (закриття рядка, що починається з лапки, інший лапками до її завершення поточної закриває лапками для поділу запиту на дві частини) було неможливо, для деяких СУБД, в тому числі, для MySQL, потрібно брати в лапки всі строкові параметри. У самому параметрі замінюють лапки на \ ", апостроф на \ ', зворотну косу риску на \\ (це називається «екранувати спецсимволи»). Це можна робити таким кодом:

```
statement := 'SELECT * FROM users WHERE name = ' + QuoteParam(userName) + ';' ;
```

Рисунок 1.13 – Приклад екранування спеціальних символів[19]

1.8 Існуючі методи виявлення зміни схеми бази даних

Після детального вивчення питання виявлення змін схеми бази даних були виявлені як загальновідомі рішення так і деякі комерційні продукти, принципи дії, яких невідомі, до чого можливо лише здогадуватися.

Існуючі загальновідомі методи полягають в наступному:

1. Використання методів баз даних

Одним з прикладів таких методів може бути **DBCC CHECKDB** для SQL Server або Azure SQL Database. DBCC CHECKDB представляє собою команду на T-SQL, яка здійснює перевірку логічної і фізичної цілісності всіх об'єктів заданої бази даних.

Але існують деякі недоліки цієї команди, як приклад, коли розміри бази даних істотно виростають, ви почнете стикатися з різними проблемами при запуску DBCC CHECKDB. Наприклад, час, необхідний для виконання процесу DBCC CHECKDB, може стати реальною перешкодою. Крім цього, може не знайтися вільного простору для знімків даних, створюваного в ході виконання DBCC CHECKDB. Іншою ж проблемою є те, що команда перевіряє цілісність

зв'язаних об'єктів, такі як первинний ключ з вторинним, тощо, а не зв'язані об'єкти не будуть перевірені.[20]

2. Використання журналу змін (транзакцій)

Журналізація змін - функція СУБД, яка зберігає інформацію, необхідну для відновлення бази даних в попереднє узгоджене стан в разі логічних або фізичних відмов.

У найпростішому випадку журналізація змін полягає в послідовній запису в зовнішню пам'ять всіх змін, які виконуються в базі даних. Записується наступна інформація:

- порядковий номер, тип і час зміни;
- ідентифікатор транзакції;
- об'єкт, що піддався зміні (номер зберігається файлу і номер блоку даних в ньому, номер рядка всередині блоку);
- попередній стан об'єкта і новий стан об'єкта.

Формована таким чином інформація називається журнал змін бази даних. Журнал містить позначки початку і завершення транзакції, і позначки прийняття контрольної точки, тому на основі аналізу цього журналу, можна виявити, які колонки були видалені, або додані, та на основі цього виявляти, чи були зміни, чи ні.[21]

3. Журнал подій

Журнали подій - перший і найпростіший інструмент для визначення статусу системи і виявлення помилок. Наприклад, в MySQL:

- Error Log - стандартний лог помилок, які збираються під час роботи сервера (в тому числі start і stop);
- Binary Log - лог всіх команд зміни БД, потрібен для реплікації і бекапов;
- General Query Log - основний лог запитів;
- Slow Query Log - лог повільних запитів.
- DDL Log — лог DDL команд.

Тому, використовуючи DDL log, можна виявити, що були внесені зміни в схему бази даних, та зробити висновки, що саме було змінено.[22]

1.9 Існуючі методи виправлення та відновлення бази даних

На сьогоднішній час існують різні підходи до відновлення бази даних:[23]

1. Manual (ручний) метод.

Ручний метод полягає в тому, що існує спеціаліст з адміністрування бази даних, який відновлює базу даних основуючись на своєму досвіді та практиці. Перевагами даного методу є те, що людина може виконати більш детальне відновлення бази даних використовуючи різні підходи та з максимальним збереженням даних. Проте, в даного методу існують і мінуси, а саме: потреба висококваліфікованого спеціаліста та затрати більшої кількості часу.

2. Відновлення з резервних копій.

Резервна копія бази даних - дублікат екземпляра або копія бази даних, яка включає в себе архітектуру та збереження даних програмного забезпечення бази даних, а відновлення, відповідно, повернення архітектури та збережених даних до стану резервної копії.

Ось деякі з методів резервного копіювання:

- Повне резервне копіювання бази даних

Повна резервна копія містить всі дані заданої бази даних або наборів файлів або файлових груп, а також журналів для забезпечення можливості подальшого відновлення цих даних.

- Диференціальне резервне копіювання

Диференціальне резервне копіювання - зберігає тільки ті зміни даних, які відбулися з часу останнього повного резервного копіювання бази даних. Коли одні й ті ж дані змінювалися багато разів з моменту останнього повного резервного копіювання бази даних, в диференціальній резервній копії зберігається остання версія змінених даних. Для цього спочатку нам потрібно відновити повну резервну копію бази даних.

- Створення резервних копій журналу транзакцій

При використанні даного підходу можна створювати резервні копії всіх подій, які відбулися в базі даних, наприклад запис кожного виконаного оператора. Це резервна копія записів журналу транзакцій і містить всі транзакції, які відбулися з базою даних. Завдяки цьому, база даних може бути відновлена до певного моменту часу. Можна навіть створити резервні копії з журналу транзакцій, якщо файли даних знищені і жодна зафіксована транзакція не втрачена.

Перевагою методу відновлення з резервних копій є те, що він вже вбудований в бази даних та його достатньо лише налаштувати. Проте метод має свої мінуси:

- Втрата даних, що були внесені в базу даних після останньої резервної копії;
- Необхідність дискового простору для копії;
- Немає гарантії того, що копія не була вже змінена злоумисником;
- Потребується спеціаліст для налаштування.

Висновки до розділу 1

У цьому розділі було розглянуто типи баз даних та їх основні поняття, цілі та галузі застосування кожного з типів, та більша увага була приділена реляційним базам даних, та мові запитів до цієї бази даних - мові структурованих запитів - SQL та пов'язаних з нею термінів. Після цього, були розглянуті основні атаки на реляційні бази даних відповідно до OWASP. Проаналізувавши літературу та веб-джерела було виявлено, що основні атаки на бази даних - це атаки на впровадження коду в SQL код та носить назву SQL injections attacks. Відповідно, до HACKING & TRICKS - SQL Injections Attacks складають 25%, від усіх мережових атак. Тому, подальший розгляд був направлений на більш детальне вивчення типів і видів цього виду атак.

Додатково, були розглянуті існуючі методи захисту від SQL injections attacks, а саме фільтрація вхідних даних, екранування, використання підготовлених запитів, відключення повідомлення про помилки, використання збережених процедур. Але усі ці методи направлені на запобігання цих типів атак, але, не існує досконалого методу, що виявляють виконані атаки, тому було більш детально вивчено це питання. Додатково, були вивчені методи виправлення виконаних атак, а саме відновлення з резервних копій та залучення спеціалістів, але кожний з підходів має відповідні мінуси, а саме втрата даних, чи додатковий спеціаліст, який буде відповідати за це. Тому, в результаті усього, були вивчені можливі підходи для виявлення та виправлення змін баз даних, особливо змін схеми бази даних, що є основною складовою реляційних баз даних, тому буде запропонований власний метод виявлення та виправлення змін схеми бази даних на основі ініціюючих скриптів, який буде позбавлений недоліків, що мають існуючі підходи.

2 АЛГОРИТМ ВИЯВЛЕННЯ ТА ВИПРАВЛЕННЯ ПОРУШЕНЬ ЦІЛІСНОСТІ СХЕМИ БАЗИ ДАНИХ

2.1 Постановка задачі

У даній дипломній роботі розглядається проблема цілісності структури бази даних, тобто, зміна схеми бази даних внаслідок несанкціонованих змін, одним з прикладів, можуть бути SQL injection атаки. Поширеність SQLIA і потенційний збиток, який вони можуть завдати, включаючи як крадіжку особистих даних так і відмову в обслуговуванні web серверу, вимагають продуктивних рішень та швидкого реагування на наявність даних атак в системі. Тому, в дипломній роботі основною задачею є аналіз існуючих методів виявлення змін в схемі бази даних та аналіз методів реагування на відновлення до початкової схеми бази даних. І власне, на основі даного аналізу, пропозиція нової методики, яка буде як виявляти порушення схеми бази даних так і пропонувати варіанти для їх вирішення.

Нашою метою є методика, яка забезпечить швидке виявлення того, що схема бази даних була змінена, відповідно до початкової ініціалізації та запропонує можливий варіант вирішення. Для цього ми будемо використовувати скрипти ініціалізації бази даних для представлення початкової схеми БД та порівнювати її з поточною схемою БД. Власне, невідповідність буде свідчити про зміни, найчастіше, несанкціоновані, та метод, що буде на основі правил пропонувати адміністратору БД ввести команди, щоб привести систему до початкової схеми. У цьому розділі ми будемо описувати загальну схему методики, яка буде складатися з таких етапів: етап представлення та аналізування початкових скриптів, етап аналізування поточного стану схеми бази даних, етап порівняння схем, етап реагування та виправлення, результатом якого буде приведення схеми до початкової схеми.

2.2 Загальна схема методики

У цьому розділі буде продемонстрована загальна схема нової методики, що є метою даної дипломної роботи, методики виявлення та виправлення порушень цілісності схеми бази даних, яка складається з певних етапів, та додатково зображена на рисунку 2.1:

1. Етап представлення та аналізування початкових скриптів.

Даний етап заключається в тому, що необхідно проаналізувати, яка схема була закладена в ініціюючий скрипт бази даних, для цього нам необхідно привести початковий скрипт схеми бази даних до спільного вигляду, до якого приведемо і поточну схему бази даних, та представити його структуру для подальшого використання.

2. Етап аналізування та представлення поточного стану схеми бази даних.

Метою на даному етапі є проаналізувати поточний стан схеми бази даних і представити поточну схему бази даних у такому ж вигляді, як і початковий скрипт бази даних, для їх подальшого порівняння.

3. Етап порівняння схем.

Етап порівняння полягає у тому, що необхідно запропонувати один із варіантів алгоритму порівняння початкової та поточної схеми бази даних на відповідність по заданим критеріям. Даний алгоритм виводить результати про відповідності. Якщо схеми не співпадають, що у більшості випадків свідчить про несанкціоновані зміни, ми переходимо до наступного етапу, а саме: етап реагування та виправлення.

4. Етап реагування та виправлення.

Останнім є етап, який, у разі якщо при порівнянні, виявиться що схеми не співпали, запропонує варіант виправлення. Якщо ж, схеми співпадуть, то даний етап не буде використаний. Даний етап є доповнюючим до всього диплому і заключним у всій схемі.

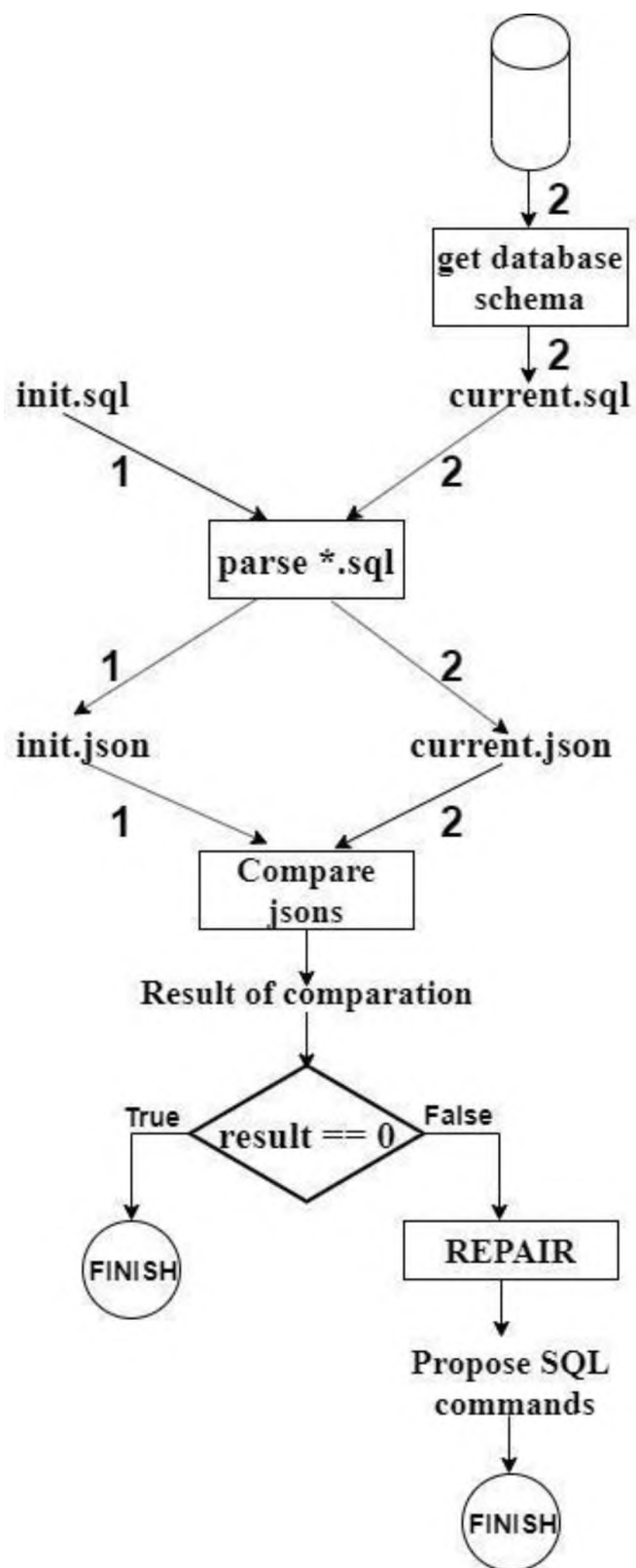


Рисунок 2.1 – Схематична реалізація методики

2.3 Етап представлення та аналізування початкових скриптів

Спочатку, невід'ємною складовою - є етап, який складається з аналізування початкових скриптів та представлення його до певного виду, який можна буде з легкістю порівняти.

Початковий скрипт БД – набір SQL команд, при запуску якого, будується схема реляційної бази даних, тому є декілька варіантів представлення початкового скрипта у вигляді початкової схеми:

1. Виконання скриптів
2. Аналіз текстових команд

Недоліком першого варіанту є те, що для виконання скрипта потрібно деяке налаштоване середовище, наприклад, тестова база даних. В випадку з варіантом 2 ніяких додаткових середовищ не потрібно, в чому і є своя перевага.

SQL (structured query language) - декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних, що складається з[24]:

- DDL (Data Definition Language) — робота зі структурою бази,
- DML (Data Manipulation Language) — робота з рядочками,
- DCL (Data Control Language) — робота з правами,
- TCL (Transaction Control Language) — робота з транзакціями.

Оскільки, нас цікавить створення структури бази, тому будемо розглядати DDL. DDL - це сімейство комп'ютерних мов, що використовуються в комп'ютерних програмах або користувачами баз даних для опису структури даних. DDL мають свою функціональну здатність, організовану за початковим словом в заяві (запит), яке майже завжди є дієсловом. У випадку з SQL:

- Create (Створити)
- Alter (Змінити)
- Drop (Видалити)

DDL, як і SQL – мови зі строгою статичною типізацією та які описані в наступних діалектах: SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2006, SQL:2008[1], SQL:2011, SQL:2016, тому зачасто незалежать від конкретної СУБД, що є ще однією перевагою.

Розберемо наступний приклад SQL-скрипта[25]:

```
1  mysql> CREATE TABLE pet (name VARCHAR(20), owner VARCHAR(20),  
2      species VARCHAR(20));
```

Рисунок 2.2 – Приклад SQL команди[25]

Даний скрипт створює таблицю з назвою “pet”, в якому будуть наступні колонки:

- name, яке відповідає ім’ям тварини, в якого тип даних VARCHAR(20), тобто текстовий тип, з максимальною довжиною 20
- owner, яке відповідає ім’ям господаря, в якого тип даних VARCHAR(20), тобто текстовий тип, з максимальною довжиною 20
- species, яке відповідає ім’ям господаря, в якого тип даних VARCHAR(20), тобто текстовий тип, з максимальною довжиною 20

Виконавши цю команду можна перевірити виконання наступною командою:

DESCRIBE <table name>, де <table name> - ім’я таблиці, в нашому випадку “pet”.

```

1  mysql> DESCRIBE pet;
2
3  +-----+-----+-----+-----+-----+
4  | Field      | Type          | Null | Key | Default | Extra |
5  +-----+-----+-----+-----+-----+
6  | name       | varchar(20)   | YES  |     | NULL    |       |
7  | owner      | varchar(20)   | YES  |     | NULL    |       |
8  | species    | varchar(20)   | YES  |     | NULL    |       |
9  +-----+-----+-----+-----+-----+

```

Рисунок 2.3 – Приклад виконання програми[25]

Як можемо побачити на малюнку і справді, була створена таблиця з трьома колонками та наступними типами даних.

Тому, було вирішено створити JSON файл, який буде відповідати схемі бази даних відповідно до початкового скрипта та поточної бази даних. Пропонується наступна структура JSON файлу:

```

1  {
2    "pet":
3    {
4      "name": "VARCHAR(20)",
5      "owner": "VARCHAR(20)",
6      "species": "VARCHAR(20)"
7    }
8  }

```

Рисунок 2.4 – Приклад JSON файлу

Це можливо із-за строгої статичної типізації. Наприклад, після слів CREATE TABLE завжди йде назва таблиці, що створюється. Після назви таблиці в дужках ідуть ім'я колонки та її тип даних, що перераховуються через

кому, тому є можливим створити програмну реалізацію, яка буде перетворювати SQL скрипт до виду JSON файлу, яка буде реалізована на PYTHON3 шляхом звичайної роботи з текстом, та вбудованої бібліотеки JSON.

2.4 Етап аналізування та представлення поточного стану схеми бази даних

2.4.1 Існуючі способи отримання поточного стану схеми бази даних

Є декілька способів, як можна отримати поточний стан схеми бази даних[26]:

1. Використання MySQLDump

Mysqldump - це утиліта командного рядка для резервного копіювання бази даних, створена Oracle. Утиліта надається як частина пакету MySQL Server.

Наступна команда може бути використана для експорту структури схеми з використанням mysqldump:



```
mysqldump --xml --no-data -h localhost -u root -pPassword schema_name > path/to/dump/file
```

Рисунок 2.5 – Приклад використання команди mysqldump[26]

Обов'язково необхідно змінити наступні параметри, щоб налаштувати команду відповідно до вашого середовища:

- «Localhost» - слід замінити на ім'я вашого сервера MySQL.
- «Root» - повинен бути замінений користувачем MySQL з правами на скидання структури схеми.
- «Password» - пароль користувача MySQL.
- «schema_name» - ім'я вашої схеми.

- «path/to/dump/file» - шлях для експорту структури схеми.

2. Використання phpMyAdmin

phpMyAdmin - це безкоштовний програмний інструмент, написаний на PHP, призначений для адміністрування MySQL через Інтернет.

Для того, щоб експортувати структуру схеми за допомогою phpMyAdmin необхідно виконати наступні кроки:

- У лівому меню вибрати назву вашої бази даних.
- На правій панелі виберіть «Експорт» у верхньому меню.
- У списку «Формат» виберіть «XML».
- З опцій методу експорту виберіть Custom - показати всі опції
- Зняти прапорець «Export Contents» в розділі «Data dump options» та натиснути Go

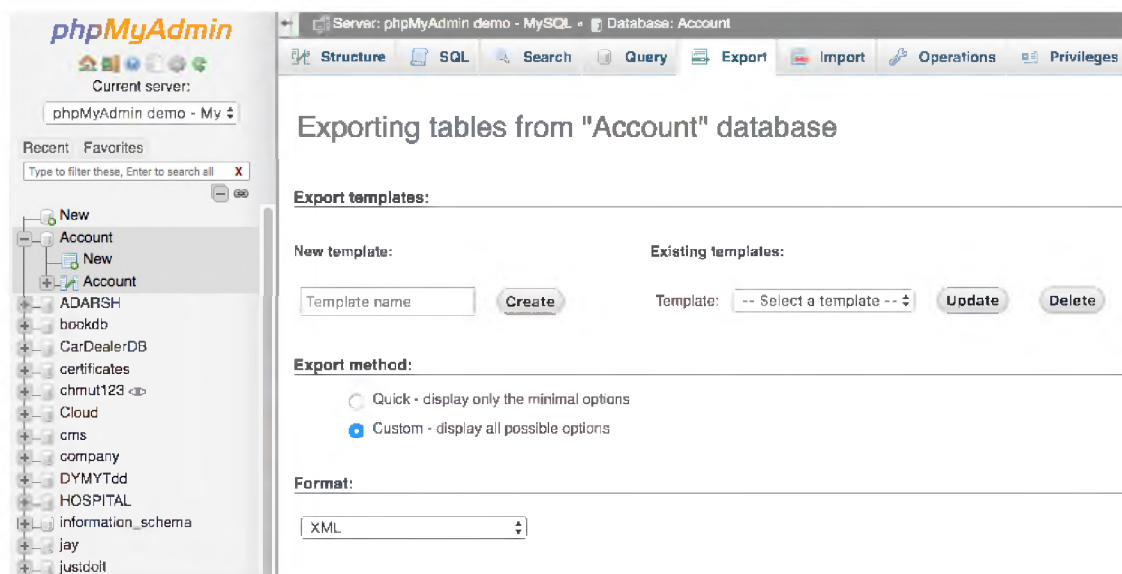


Рисунок 2.6 – Приклад використання phpMyAdmin[26]

3. Використання Sequel Pro

Sequel Pro - це застосунок для управління базами даних Mac для роботи з базами даних MySQL.

Кроки для того, щоб експортувати структуру схеми бази даних за допомогою Sequel Pro:

- У меню «Файл» вибрати «Експорт».
- У верхньому меню вибрати SQL в якості методу експорту.
- Зняти прапорець з опції: Зміст
- Зняти прапорець з опції: DROP TABLE syntax
- Зняти прапорець з опції «Output BLOB as Hex»
- натиснути Експорт

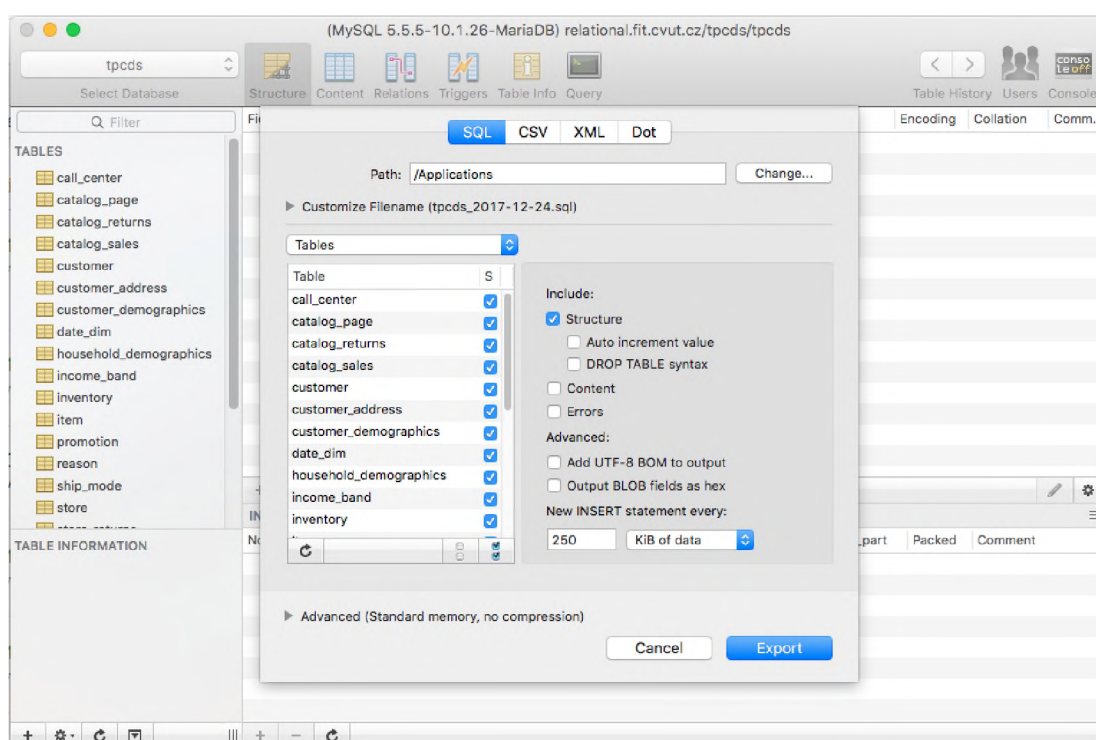


Рисунок 2.7 – Приклад використання Sequel Pro[26]

4. Використання MySQL Workbench

MySQL Workbench - це уніфікований візуальний інструмент для архітекторів, розробників і адміністраторів баз даних. MySQL Workbench надає моделювання даних, розробку SQL і комплексні інструменти адміністрування для налаштування сервера, адміністрування користувачів, резервного

копіювання та багато чого іншого. MySQL Workbench доступний в Windows, Linux і Mac OS X.

Кроки для експорту структури схеми з використанням MySQL Workbench:

В меню «Сервер» виберіть «Експорт даних» ⇒ З лівого боку виберіть базу даних для експорту ⇒ Виберіть «Dump structure only» як dump метод ⇒ Зніміть прапорці з опцій: Dump Stored Procedures and Functions, Dump Events, Dump Triggers ⇒ У розділі «Параметри експорту» виберіть «Експорт в автономний файл» ⇒ Натисніть Почати експорт

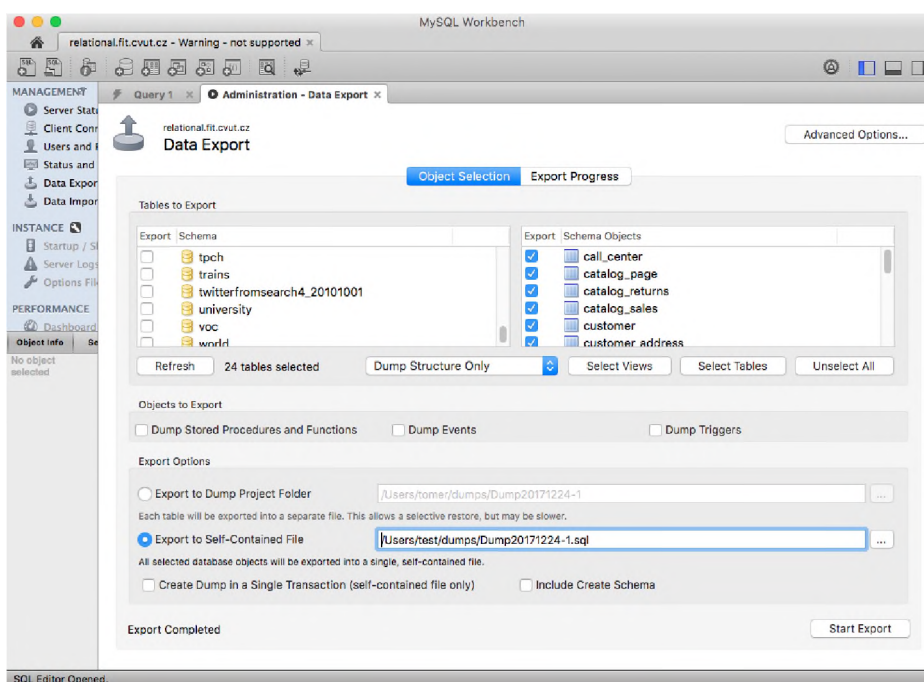


Рисунок 2.8 – Приклад використання MySQL Workbench[26]

5. Використання HeidiSQL

HeidiSQL - це безкоштовний інструмент адміністрування з відкритим вихідним кодом для MySQL і його форків, а також Microsoft SQL Server і PostgreSQL.

Необхідно виконати такі дії для експорту структури схеми бази даних з використанням HeidiSQL:

У меню «Сервіс» виберіть «Експортувати базу даних як SQL» ® З лівого боку виберіть базу даних для експорту ® Зніміть прапорець «Створити» і «Видалити» поруч з Databases ® Зніміть прапорець «Видалити» поруч з таблицями ® Встановіть прапорець «Створити» поруч з таблицями ® Виберіть Інформація відсутня, щоб витягти тільки структуру схеми ® Виберіть один файл SQL в якості методу виведення ® натисніть Експорт

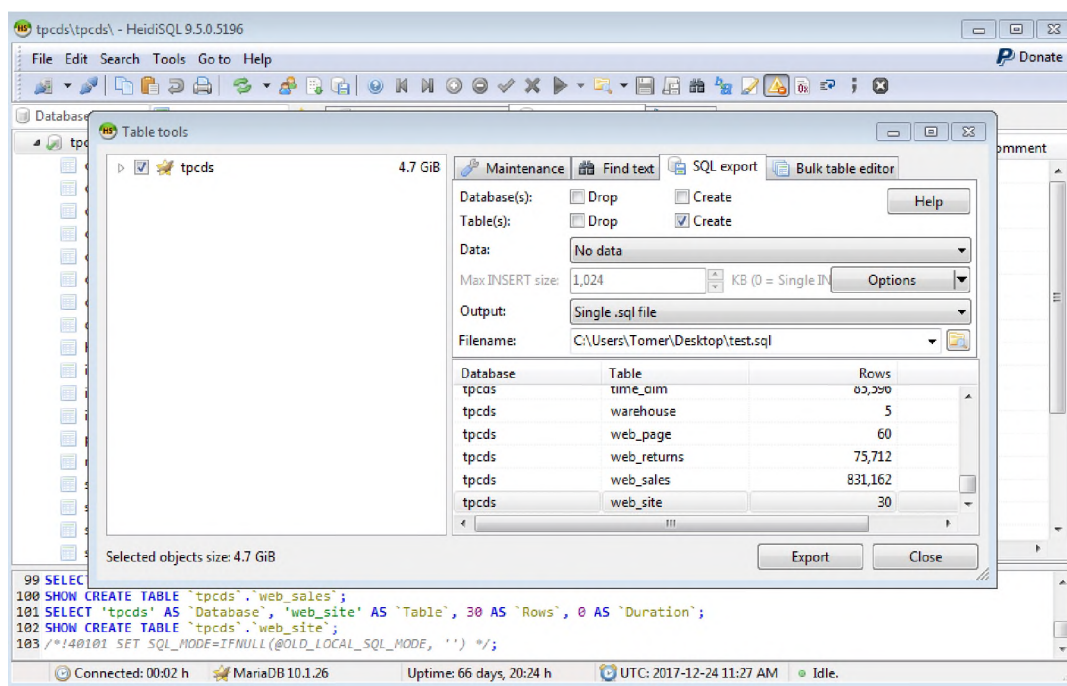


Рисунок 2.9 – Приклад використання HeidiSQL[26]

6. Використання SQLYog

SQLyog - це інструмент з графічним інтерфейсом для СУБД MySQL. Розроблено Webuyog.

Для того, щоб експортувати структуру схеми з використанням SQLYog необхідно:

У меню Сервіс виберіть Резервна копія бази даних як дамп SQL ® У верхній панелі виберіть «Експортувати як SQL: тільки структура» ® З лівого боку виберіть базу даних для експорту ® З лівого боку зніміть всі типи об'єктів, крім таблиць ® Зніміть всі опції на правій бічній панелі ® Натисніть Експорт

7. Використання DbForge Studio Express

Безкоштовний клієнт MySQL, який надає основні функції для розробки баз даних. Він включає в себе редактор об'єктів бази даних, інструменти настройки безпеки, розширений редактор даних і експорт / імпорт з CSV і ODBC.

Необхідно виконати наступні кроки, щоб експортувати структуру схеми бази даних за допомогою DbForge Studio Express:

В меню База даних виберіть Резервне копіювання і відновлення => Резервне копіювання бази даних => Натисніть Далі, щоб перейти до екрану «Створення резервної копії вмісту» => Виберіть опцію структури і зніміть галочку з опції Data => Зніміть прапорці з усіх типів об'єктів, крім таблиць => Натисніть Резервне копіювання.

8. Використання pg_dump

pg_dump - це популярна утиліта для резервного копіювання бази даних PostgreSQL.

Наступна команда може бути використана для експорту структури схеми з використанням pg_dump:

```
1 | pg_dump -s databasename > schema.sql
```

Рисунок 2.10 – Приклад використання команди pg_dump[26]

Звичайно, потрібно переконатися, що ви встановили ваше ім'я бази даних у наведеній вище команді перед її виконанням.

9. Використання SQL Server Management Studio

SQL Server Management Studio (SSMS) - це інтегроване середовище для управління будь-якими базами даних SQL Server. SSMS надає інструменти для настройки, моніторингу та адміністрування екземплярів бази даних SQL Server.

Необхідні кроки для експорту структури схеми з використанням SSMS:

На лівій панелі клацніть правою кнопкою миші базу даних, для якої буде експортуватися структура схеми => Виберіть Завдання => виберіть Створити сценарії => Натисніть Далі на екрані вітання => Натисніть «Далі» на екрані «Вибір об'єктів бази даних для сценарію» => Натисніть Advanced => виберіть опцію «Типи даних для сценарію» і виберіть «Тільки схема» => Виберіть шлях для експорту файлу і натисніть «Далі» => Натисніть Готово.

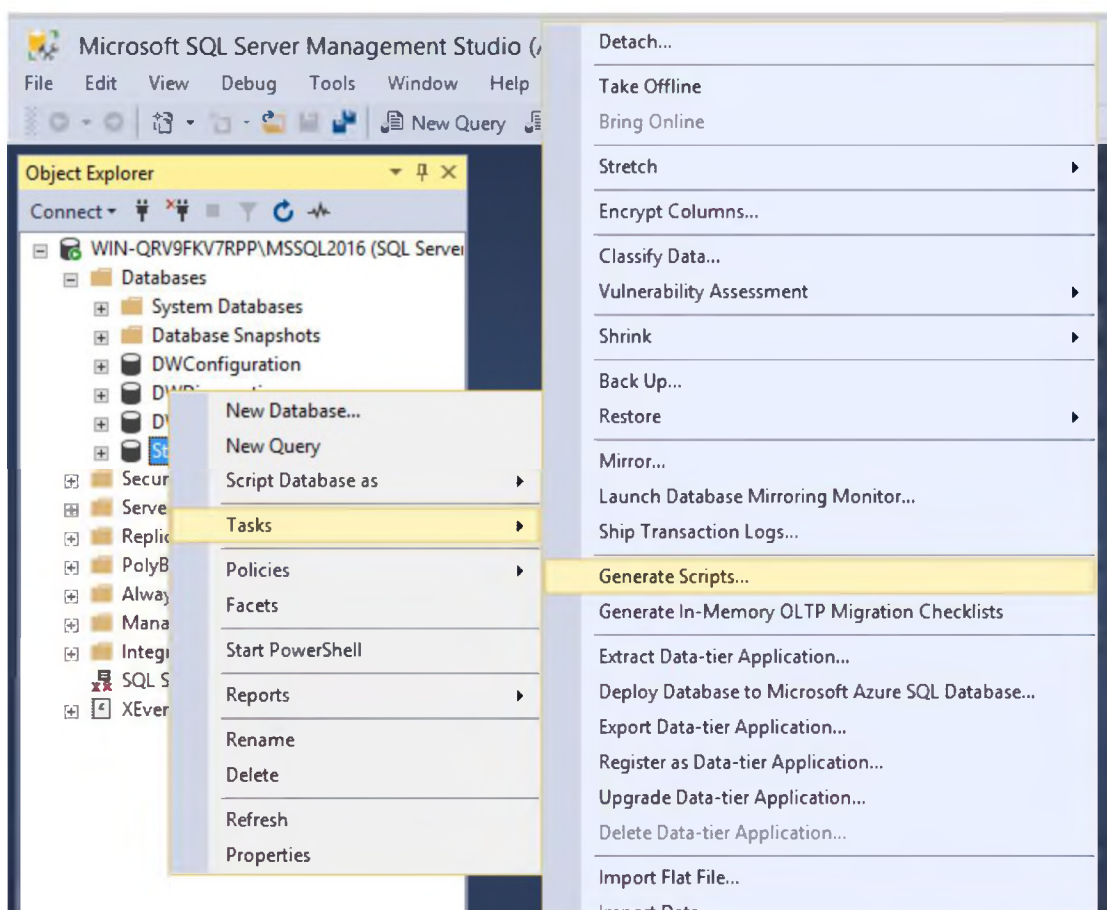


Рисунок 2.11 – Приклад використання SQL Server Management Studio[26]

10. Використання SQL команд

Одним із способів опису схеми таблиць є SQL команда DESCRIBE, яка застосовується для MySQL.

Іншим способом представлення поточного стану схеми бази даних є використання SQL команди sp_columns, яка застосовується для MS SQL.

Команда DESCRIBE відображає структуру таблиць та інформацію про кожний зі стовбців таблиці.

Нижче наведено приклад виконання даної команди:

DESCRIBE <table name>, де <table name> - ім'я таблиці, в нашому випадку "pet".

```

1  mysql> DESCRIBE pet;
2
3  +-----+-----+-----+-----+-----+
4  | Field  | Type      | Null | Key | Default | Extra |
5  +-----+-----+-----+-----+-----+
6  | name   | varchar(20) | YES  |     | NULL    |       |
7  | owner  | varchar(20) | YES  |     | NULL    |       |
8  | species | varchar(20) | YES  |     | NULL    |       |
9  +-----+-----+-----+-----+-----+

```

Рисунок 2.12 – Приклад виконання команди DESCRIBE[26]

- «Field» - вказує ім'я стовпця;
- «Type» - є типом даних для стовпця;
- «NULL» - вказує, чи може стовпець містити значення NULL;
- «Key» - вказує, чи індексується стовпець;
- «Default» - вказує значення за замовчуванням для стовпця;
- «Extra» - відображає спеціальну інформацію про стовпець: якщо стовпець був створений з параметром AUTO_INCREMENT, значення буде auto_increment, а не порожнє.

2.4.2 Опис етапу аналізування та представлення поточного стану схеми бази даних

Даний етап є наступним після етапу представлення та аналізування початкових скриптів. На цьому етапі необхідно проаналізувати поточний стан схеми бази даних і представити поточну схему бази даних у такому ж вигляді, у якому буде представлений початковий скрипт, для їх подальшого порівняння.

З усіх вище продемонстрованих способів у пункті 2.4.1 можна зробити висновок, що існує багато способів як отримати поточний стан схеми бази даних. Деякі варіанти мають графічний інтерфейс, деякі – консольні застосунки. Основуючись на наші потреби, було вирішено, що нам підійдуть варіанти отримання поточної схеми бази даних через командний рядок або через SQL команду DESCRIBE, оскільки це буде з легкістю зробити через мову програмування Python3 та з легкістю зв'язати з іншими модулями нашої методики. Із усіх перелічених варіантів, залишаються наступні:

- Mysqldump;
- pg_dump;
- SQL команда describe;
- sp_columns.

Проаналізувавши властивості кожної з них, було виділено, що SQL команди DESCRIBE та sp_columns потребують додаткових дій з аналізу поточного стану з поля виводу, а команди Mysqldump та pg_dump формують SQL скрипт аналогічний скрипту ініціалізації, що призводить до того, щоб привести скрипт ініціалізації та поточний SQL скрипт, потрібно виконати однакові дії, а саме дії запропоновані в розділі 2.3. Оскільки, Mysqldump працює з MySQL, а pg_dump з PostgreSQL, тому потрібно відслідковувати, з якою базою даних потрібно працювати та викликати один чи інший застосунок, але робота - наукова, та мета її – запропонувати ідею тому зупинимось лише на використанні pg_dump та базою даних PostgreSQL.

Результатом усіх дій буде: отримання скрипту поточного стану -> використання методу аналізування та представлення скриптів (п.2.3) -> отримуємо JSON з поточною структурою, який з легкістю можна порівняти та зробити висновок про цілісність структури бази даних, тому можна переходити до наступного етапу, а саме: порівняння схем.

```
1 {  
2   "pet":  
3   {  
4     "name": "VARCHAR(20)",  
5     "owner": "VARCHAR(20)",  
6     "species": "VARCHAR(20)"  
7   }  
8 }
```

Рисунок 2.13 – Приклад JSON файлу для поточної структури

2.5 Етап порівняння схем

Етап порівняння схем є важливим, адже на основі нього можна робити висновки про існування несанкціонованих змін та про подальші дії у методиці. Суть даного етапу полягає у тому, що необхідно запропонувати один із варіантів алгоритму порівняння початкової та поточної схеми бази даних на відповідність по заданим критеріям. Після порівняння, алгоритм виводить результати про відповідності. Якщо схеми не співпадають, що у більшості випадків свідчить про несанкціоновані зміни, ми переходимо до наступного етапу, а саме: методу реагування та виправлення.

Алгоритм порівняння в нашому випадку буде полягати в наступному:

1. Зчитування JSON файлу, який відповідає схемі бази даних відповідно до початкового скрипту в список.

2. Зчитування JSON файлу, що відповідає поточній схемі бази даних в список.
3. Порівняння списків шляхом віднімання списку з поточною схемою від списку з початковою схемою бази даних.
4. Отримання результату та його вивід у консоль у вигляді повідомлення про відповідність чи невідповідність даних схем:
 - а. Якщо схеми співпадають, то у консоль виводиться повідомлення, що невідповідностей не виявлено.
 - б. Якщо схеми не співпадають, то у консоль виводиться повідомлення, що знайдені невідповідності у вигляді:
 - Зайва таблиця <ім'я таблиці>;
 - Зайва колонка <ім'я колонки>;
 - Невідповідність типів колонки <ім'я колонки>.

Представимо даний алгоритм у вигляді схеми, що зображена нижче:

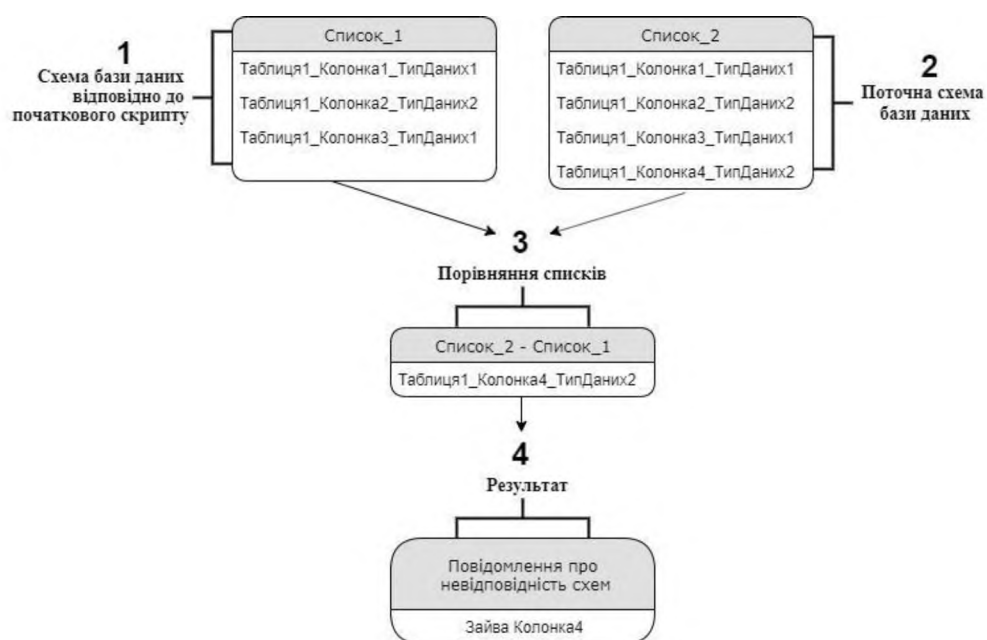


Рисунок 2.14 – Схема роботи алгоритму порівняння схем

Отже, якщо при порівнянні схем, змін не виявлено, то база даних знаходиться в тому ж стані, що і була в скрипті ініціалізації. Звідси слідує, що дій щодо виправлення схеми БД не потрібно. Якщо ж схеми не співпали і алгоритм видав невідповідності у них, то переходимо до етапу реагування та виправлення.

2.6 Етап реагування та виправлення

Даний етап є останнім етапом у розробці методики та базується на результатах попереднього етапу. Він застосовується лише у тому випадку, якщо при порівнянні схем, виявиться, що вони не співпали. Основною ідеєю його являється реагування на невідповідність схем бази даних і пропозиція виправлення даного порушення, яке в більшості випадків свідчить про несанкціоновані зміни.

У розділі 1 було розглянуто та проаналізовано існуючі підходи до відновлення бази даних та оскільки кожен містить певні недоліки, в даній дипломній роботі буде запропонований новий метод, який буде пропонувати команди для адміністратора бази даних, для відновлення схеми бази даних до початкового стану.

Принцип роботи полягає у наступному: після порівняння схем отримуємо які таблиці/колони/типи даних не співпадають з початковою схемою, та алгоритмом на основі набору правил пропонуємо команди. Прикладом може бути наступна невідповідність:



Рисунок 2.15 – Приклад невідповідності схем бази даних

Отже, в отриманій різниці можна побачити, що існує Таблиця1_Колонка4_ТипДаних2, чого не існує в початковій схемі, тому метод буде пропонувати наступну команду:

ALTER TABLE “Таблиця1” DROP “Колонка4”.

Тому, після виконання цієї команди схема бази даних перейде у відповідність з початковою схемою, відповідно до скрипта ініціалізації.

Висновки до розділу 2

У цьому розділі була поставлена задача відносно теми магістерської дисертації та описана мета роботи. Також, у цьому розділі було продемонстровано загальну схему методики виявлення та виправлення порушень цілісності схеми бази даних, яка складається з 4 етапів, а саме: етапу представлення та аналізування початкових скриптів, етапу аналізування та представлення поточного стану схеми бази даних, етап порівняння схем та етапу реагування та виправлення, та продемонстровано схематичну реалізацію даної методики для більш кращого та чіткого розуміння принципу її роботи.

Було розібрано кожен із даних етапів, де покроково описано та проаналізовано принцип їх роботи, механізми, як MySQL dump, pg_dump для отримання поточної схеми бази даних, та інші механізми, що використовуються у етапах методики. Додатково, запропоновані алгоритми для аналізу скриптів, порівняння списків, які будуть програмно реалізовані в практичній частині магістерської дисертації та будуть наведені результати роботи даної методики.

3 МЕТОДИКА ВИЯВЛЕННЯ ТА ВИПРАВЛЕННЯ ПОРУШЕНЬ ЦІЛІСНОСТІ СХЕМИ БАЗИ ДАНИХ

Після аналізу впливу SQL атак на можливі зміни в схемі реляційної бази даних, аналізу існуючих методів виявлення порушень цілісності схем та методів реагування були знайдені недоліки та був запропонований алгоритм виявлення змін схеми, основуючись на початкові скрипти, та, додатково запропонований алгоритм виправлення змін, тобто приведення схеми до стану початкового скрипта. У цьому розділі описується детальна програмна реалізація, взаємодія між компонентами, послідовність функціональних викликів. Методика була реалізована на скриптовій мові програмування Python3. Оскільки, робота наукова, тому будемо працювати з командою `pg_dump`, для отримання скрипту з поточною схемою для PostgreSQL, але методика легко масштабується і для інших баз даних, наприклад, додаванням використання `mysql_dump`, для роботи з MySQL, тощо.

3.1 Розробка методики виявлення та виправлення порушень цілісності схеми бази даних

На рисунку 3.1 зображене загальне представлення методики, в якому виконується огляд всієї системи, виявлення основних компонентів, які будуть розроблені для продукту. Дане представлення коротко описує всі платформи, системи, та їх взаємодію. На даному рисунку можна побачити, що застосунок консольний та користувач взаємодіє з DetectAndRestore тільки через термінал. Сам модуль реалізований на Python3.

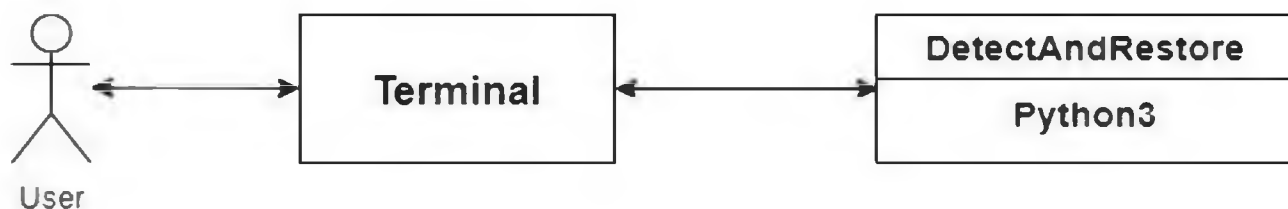


Рисунок 3.1 - Загальне представлення методики виявлення та виправлення порушень схеми БД

На рисунку 3.2 зображена діаграма послідовностей для запропонованої методики. Діаграма послідовностей відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовності відправлених повідомлень. На діаграмі показані у вигляді вертикальних ліній різні процеси або об'єкти, що існують водночас. Надісланні повідомлення зображуються у вигляді горизонтальних ліній, в порядку відправлення. Як видно з рисунку, ми маємо 4 модуля, які взаємопов'язані та об'єднуються у 1 головний модуль нашої програми.

- На вході ми маємо скрипт ініціалізації `init.sql`
- Далі у роботу вступає об'єкт послідовності `DatabaseSchemaExtractor`, який викликає метод `extractPGschema()` і на виході отримує поточну схему бази даних `current.sql`.
- Наступним об'єктом є `SQLParser`, який викликає метод `parseSQL()` для обох схем бази даних. Даний метод перетворює SQL скрипти до виду JSON файлів і опісля ми отримуємо два файли `init.json` та `current.json`.
- Наступний модуль - `Comparator`, задачею якого є порівняти два JSON файли і видати результат порівняння. На вхід даний модуль приймає два JSON файли `init.json` та `current.json` та зчитує їх у список для порівняння шляхом віднімання списку з поточною схемою від списку з початковою схемою бази даних, викликаючи метод `compareJSONs()`. На виході отримуємо результат у вигляді списку невідповідностей та виводимо

результат у консоль у вигляді повідомлення про відповідність чи невідповідність даних схем.

- Останній об'єкт Restorer є додатковим до усього методу. Даний модуль буде викликаний лише у тому випадку, якщо при порівнянні схем, виявиться, що вони не співпали. На вхід він приймає список невідповідностей та застосовуючи метод `restore()` видає відповідні команди для відновлення схеми бази даних до початкового стану.

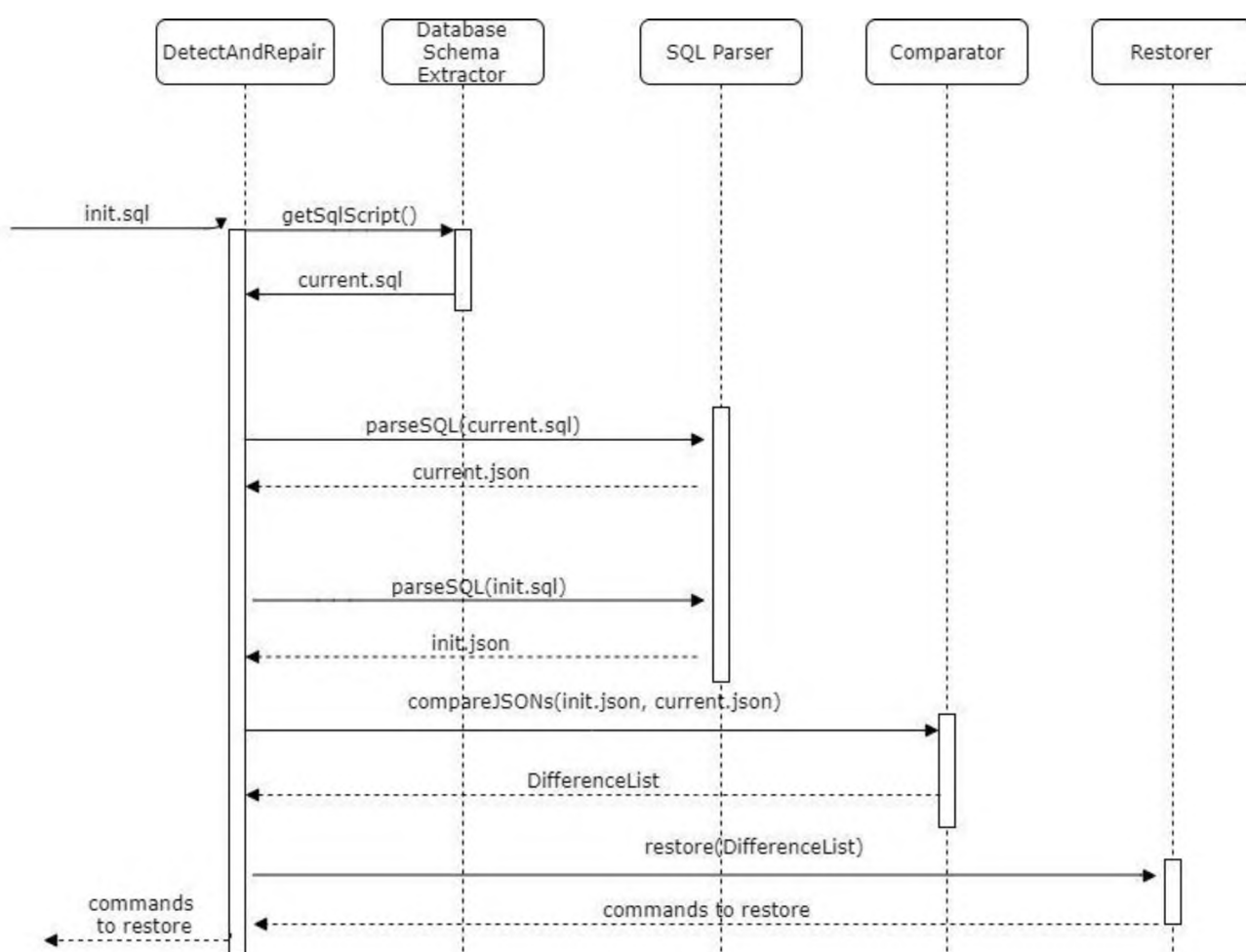
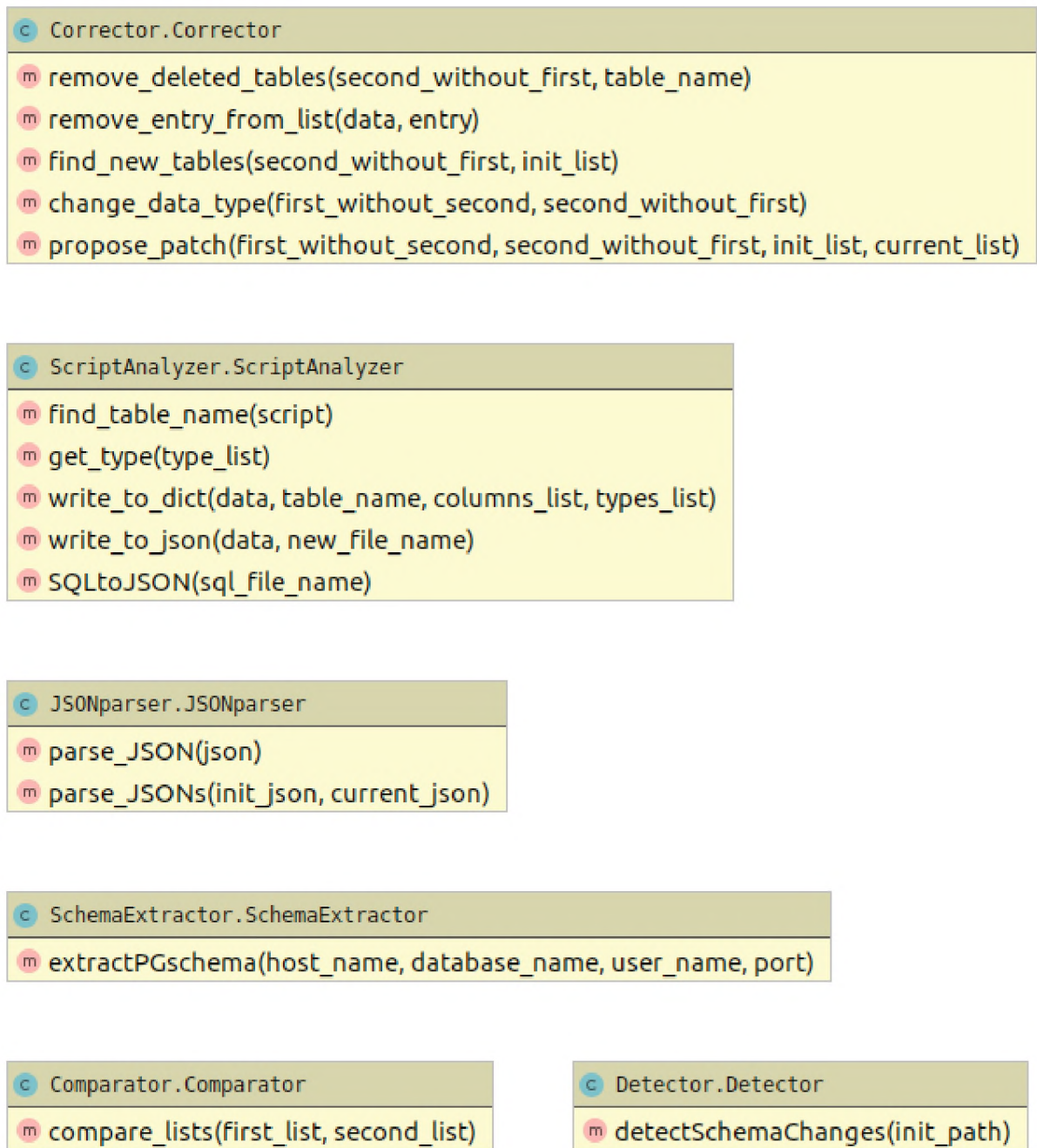


Рисунок 3.2 – Діаграма послідовностей для методики

На рисунку 3.3 зображена діаграма класів для розробленого застосунку. Діаграма класів — статичне представлення структури моделі. Вона відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма виконана відповідно до Unified Modeling Language (UML).

У діаграмі класів можна побачити, що застосунок має наступні класи: `Corrector`, `ScriptAnalyzer`, `JSONparser`, `Detector`, `Comparator`, `SchemaExtractor`. Клас `Corrector` відповідає за аналіз різниці списків, та на основі правил пропонує команди, які потрібно ввести для виправлення схеми бази даних. Клас `ScriptAnalyzer` відповідає за аналіз *.sql файлів та запису *.json файлу. Клас `JSONparser` відповідає за відкриття JSON файлу та отримання списку з елементами у вигляді Таблиця1----Колонка4----ТипДаних2. Наступний клас `Comparator` - він порівнює списки, отримані з `JSONparser`. Клас `SchemaExtractor` відповідає за отримання поточної схеми бази даних за допомогою виклику `pg_dump`. Останній клас `Detector` - він використовує попередньо описані класи, та видає результат, чи є факт втручання в схему бази даних.



Powered by yFiles

Рисунок 3.3 – Діаграма класів для методики

3.2 Програмна реалізація

У цьому розділі роздивимось більш детально програмну реалізацію та програмний код.

На рисунку 3.4 зображений програмний код D&R.py. Як можемо побачити, він використовує два класи, це Detector для отримання результатів чи є зміни в схемі бази даних та Corrector, у випадку, якщо зміни були виявлені тоді запропонувати варіант вирішення. За це відповідає змінна condition.

```

1 import sys
2
3 from Corrector import *
4 from Detector import *
5
6 if __name__ == '__main__':
7     condition, first_without_second, second_without_first, init_list, current_list = Detector.detectSchemaChanges(
8         sys.argv[1])
9     if condition:
10         Corrector.propose_patch(first_without_second, second_without_first, init_list, current_list)
11

```

Рисунок 3.4 – Програмний код D&R.py

Роздивимось спочатку більш детально клас Detector, що зображений на рисунку 3.5. Як можемо побачити, на 9-ому рядку ми отримуємо json файл з початкового скрипту за допомогою класу ScriptAnalyzer. Далше должна быть получение текущей схемы бд. На 12-ому рядку отримуємо json файл з скрипту поточної бази даних. На 14-ому рядку використовується клас JSONparser, який отримує JSON файли та перетворює їх на списки заданого формату. На 16-ому рядку використовується клас Comparator, який порівнює задані списки, та повертає інші 2 списки, а саме список, в якому є те, чого немає в іншому та навпаки, та на основі цих результатів виводиться результат. Результат полягає в наступному: якщо списки не ідентичні, а саме, існують якісь записи, що є в початковому скрипті, але немає в поточному, або навпаки, тому робиться висновок, що база даних була змінена. Якщо ж, списки ідентичні, тобто різниця одного від іншого не дала результатів, тому робиться висновок, що схема бази даних не була змінена.


```

1 from SchemaExtractor import *
2 from Comparator import *
3 from JSONparser import *
4 from ScriptAnalyzer import *
5
6
7 class Detector:
8     @staticmethod
9     def detectSchemaChanges(init_path):
10         init_json = ScriptAnalyzer.SQLtoJSON(init_path)
11
12         second_path = SchemaExtractor.extractPGSchema("localhost", "testDB", "root", "5432")
13         current_json = ScriptAnalyzer.SQLtoJSON(second_path)
14
15         init_list, current_list = JSONparser.parse_JSONs(init_json, current_json)
16
17         first_without_second, second_without_first = Comparator.compare_lists(init_list, current_list)
18
19         if len(first_without_second) != 0 or len(second_without_first) != 0:
20             print("Some problems with database!!!")
21             print("Problems with:\n" + str(first_without_second) + " " + str(second_without_first) + "\n")
22             return True, first_without_second, second_without_first, init_list, current_list
23         else:
24             print("Problems not found")
25             return False, [], [], [], []
26
27

```

Рисунок 3.5 – Програмний код класу Detector

Опишемо більш детально допоміжні класи до класу Detector. Та почнемо з ScriptAnalyzer. Основною функцією є SQLtoJSON, яка приймає на вхід шлях до *.sql файлу. Як видно з рисунку, на 37-й строчці, виконується розділення усього файлу на окремі команди, шлях знаходження “;”, оскільки згідно до синтаксису мови SQL, кожна команда виділяється цим знаком. Наступний етап - це циклічний прохід по кожній з команд скрипта, в якому знаходиться алгоритм знаходження імені таблиці, імен колонок та їх типів даних. В кінці-кінців, усе це зберігається в словник, та врешті-решт записується до файлу.

```

33     @staticmethod
34     def SQLtoJSON(sql_file_name):
35         data = {}
36         with open(sql_file_name, "r") as f:
37             scripts = f.read().replace("\n", "").split(";")[:-1]
38             for script in scripts:
39                 table_name = ScriptAnalyzer.find_table_name(script)
40
41                 columns_list = list()
42                 types_list = list()
43                 columns = script[script.find("(") + 1:]
44                 columns = columns.split(",")
45
46                 for column in columns:
47                     if "PRIMARY" in column:
48                         continue
49                     if "NOT NULL" in column or "UNIQUE" in column:
50                         column = column[:column.find("NOT NULL")]
51                     if "UNIQUE" in column:
52                         column = column[:column.find("UNIQUE")]
53
54                     column_split = column.strip().split(" ")
55                     columns_list.append(column_split[0])
56                     types_list.append(ScriptAnalyzer.get_type(column_split[1:]))
57
58                 ScriptAnalyzer.write_to_dict(data, table_name, columns_list, types_list)
59
60         ScriptAnalyzer.write_to_json(data, sql_file_name.split(".")[0])
61         return data
62

```

Рисунок 3.6 – Программний код класу ScriptAnalyzer

На рисунку 3.7 зображений програмний код класу SchemaExtractor. Як видно, з програмного коду, дана функція запускає термінал з командою `pg_dump` з спеціальними параметрами. Додатково можуть бути реалізовані функції, які отримують схему бази даних з інших реляційних баз даних.

```

1  from subprocess import PIPE, Popen
2
3
4  class SchemaExtractor:
5      @staticmethod
6      def extractPGSchema(host_name, database_name, user_name, port):
7          command = 'pg_dump -h {0} -d {1} -U {2} -p {3} --schema-only -f ./current.sql' \
8                  .format(host_name, database_name, user_name, port)
9          p = Popen(command, shell=True, stdin=PIPE, stdout=PIPE, stderr=PIPE)
10         p.communicate()
11         return "current.sql"
12
13

```

Рисунок 3.7 – Программний код класу SchemaExtractor

На рисунку 3.8 зображений клас JSONparser, що перетворює JSON файл до списку зі специфічною структурою, а саме на 5-7 строчках рисунку.

```

1 class JSONparser:
2     @staticmethod
3     def parse_JSON(json):
4         init_list = list()
5         for key in json.keys():
6             for column_name in json[key].keys():
7                 init_list.append(str(key) + "----" + str(column_name) + "----" +
8                                 str(json[key][column_name]))
9
10        return init_list
11
12    @staticmethod
13    def parse_JSONs(init_json, current_json):
14        return JSONparser.parse_JSON(init_json), JSONparser.parse_JSON(current_json)

```

Рисунок 3.8 – Програмний код класу JSONparser

На рисунку 3.9 зображений клас Comparator, в якому порівнюються списки, що були отримані в результаті виклику функції parse_JSONs(). Результатом виконання є перший список, що не містить елементів другого та другий список, що не містить елементів першого.

```

1 class Comparator:
2     @staticmethod
3     def compare_lists(first_list, second_list):
4         first_without_second = sorted(set(first_list) - set(second_list))
5         second_without_first = sorted(set(second_list) - set(first_list))
6         return first_without_second, second_without_first
7

```

Рисунок 3.9 – Програмний код класу Comparator

Як вже і було раніше сказано, на основі аналізу цих списків робиться висновок чи поточна схема бази даних співпадає з початковою, чи ні. Якщо ж не співпадає, то викликається інший клас, який відповідає за генерацію команд, які приведуть базу даних до початкового стану, а саме Corrector.

На рисунку 3.10 зображена основна функція, propose_patch(), яка відповідає за генерацію команд. Дана функція поділяється на 4 основних блоки, а саме:

1. Знаходження зайвих таблиць
2. Знаходження колонок, в яких був змінений тип даних
3. Знаходження видалених колонок
4. Знаходження зайвих колонок

```

49 def propose_patch(first_without_second, second_without_first, init_list, current_list):
50     # Finding new tables
51     while True:
52         condition, second_without_first = Corrector.find_new_tables(second_without_first, init_list)
53         if not condition:
54             break
55
56     # Change data type
57     while True:
58         condition, first_without_second, second_without_first = Corrector.change_data_type(first_without_second,
59                                                                                             second_without_first)
60         if not condition:
61             break
62
63     # Finding deleted columns
64     if len(first_without_second) > 0:
65         print("Founded deleted columns, let's add them")
66         print("You need to execute:")
67         for item in first_without_second:
68             splitted = item.split('----')
69             print("ALTER TABLE " + splitted[0] + " ADD COLUMN " + splitted[1] + " " + splitted[2] + ";\n")
70
71     # Finding new columns
72     if len(second_without_first) > 0:
73         print("Founded new columns, let's remove them")
74         print("You need to execute:")
75         for item in second_without_first:
76             splitted = item.split('----')
77             print("ALTER TABLE " + splitted[0] + " DROP " + splitted[1] + ";\n")
78

```

Рисунок 3.10 – Програмний код функції propose_patch()

3.3 Результати роботи

Отже, в результаті роботи була розроблена методика виявлення та виправлення змін схеми бази даних та був реалізований програмний прототип, який описується в розділі 3.2.

Виявлення змін баз даних виконується на основі порівняння двох списків, які були отримані з JSON файлів в спеціальному форматі. Тобто, якщо множина об'єктів (таблиць/колонок) поточної схеми співпала з множиною об'єктів початкового скрипта, то обидва списки будуть пусті і змін не буде виявлено, в іншому випадку, це буде свідчити про зміни в схемі баз даних.

На основі виявлення був запропонований прототип виправлення схеми бази даних, тобто пропонується команда, або набір команд, які приведуть схему баз даних без відкату до резервної копії, хоча з втратою або модифікацією даних, в колонках, які були видалені.

На даному етапі розглядаються наступні вектори виправлення:

1. Знаходження зайвих таблиць
2. Знаходження колонок, в яких був змінений тип даних
3. Знаходження видалених колонок
4. Знаходження зайвих колонок

Розглянемо їх більш детально.

3.3.1 Знаходження зайвих таблиць

Одним з варіантів виправлення, який представляє прототип — є знаходження зайвих таблиць та їх видалення. Приклад скриптів зображений на рисунках 3.11 та 3.12:

```
CREATE TABLE account(
  user_id serial PRIMARY KEY,
  username VARCHAR (50) UNIQUE NOT NULL,
  password VARCHAR (50) NOT NULL,
  email VARCHAR (355) UNIQUE NOT NULL,
  created_on TIMESTAMP NOT NULL,
  last_login TIMESTAMP
);
```

Рисунок 3.11 – Приклад ініціуючого скрипта для випадку знаходження зайвих таблиць

```
CREATE TABLE account(
  user_id serial PRIMARY KEY,
  username VARCHAR (50) UNIQUE NOT NULL,
  password VARCHAR (50) NOT NULL,
  email VARCHAR (355) UNIQUE NOT NULL,
  created_on TIMESTAMP NOT NULL,
  last_login TIMESTAMP
);
CREATE TABLE added_table(
  malicious_id serial PRIMARY KEY,
  malicious_column_1 VARCHAR (50) UNIQUE NOT NULL,
  malicious_column_2 VARCHAR (50) NOT NULL,
  malicious_column_3 VARCHAR (355) UNIQUE NOT NULL
);
```

Рисунок 3.12 – Приклад поточної схеми БД для випадку знаходження зайвих таблиць

Тому, після запуску нашого застосунку отримали наступний вивід з терміналу:

```
Some problems with database!!!
Problems with:
[] ['added_table----malicious_column_1----VARCHAR(50)', 'added_table----malicious_column_2----VARCHAR(50)', 'added_table----malicious_column_3----VARCHAR(355)']

Founded new table, let's remove their
You need to execute:
DROP TABLE added_table;
```

Рисунок 3.13 – Результат виконання програми для випадку знаходження зайвих таблиць

Як видно з рисунку 3.13, наша методика виявила таблицю, яка була додана відносно початкового скрипту та запропонована команда з видалення цієї таблиці, а саме `DROP TABLE added_table;`

Програмна реалізація виглядає наступним чином:

```
12 @staticmethod
13 def find_new_tables(second_without_first, init_list):
14     for index, item in enumerate(second_without_first):
15         table_name = item.split('----')[0]
16         is_table_in_init = False
17         for init_item in init_list:
18             table_name_init = init_item.split('----')[0]
19             if table_name == table_name_init:
20                 is_table_in_init = True
21                 break
22         if not is_table_in_init:
23             print("Founded new table, let's remove their")
24             print("You need to execute:")
25             print("DROP TABLE " + table_name + ";\n")
26             return True, Corrector.remove_deleted_tables(second_without_first, table_name)
27     return False, second_without_first
```

Рисунок 3.14 – Програмна реалізація для випадку знаходження зайвих таблиць

Як можемо побачити з програмної реалізації, виконуються наступні дії: проходиться по усім зайвим об'єктам з поточної бази даних та виділяються саме імена таблиць та перевіряється, чи була створена така таблиця в початковому скрипті, та повертається статус False, якщо була, та виводяться команди та повертається статус True та видаляються всі записи з іменем цієї колонки з списку із зайвими об'єктами поточної бази даних.

3.3.2 Знаходження колонок, в яких був змінений тип даних

Інший варіант виправлення — знаходження змінених типів даних. Для прикладу приведемо наступні скрипти, що показані на рисунках 3.15 та 3.16:

```
CREATE TABLE account(
    user_id serial PRIMARY KEY,
    username VARCHAR (50) UNIQUE NOT NULL,
    password VARCHAR (50) NOT NULL,
    email VARCHAR (355) UNIQUE NOT NULL,
    created_on TIMESTAMP NOT NULL,
    last_login VARCHAR (355)
);
```

Рисунок 3.15 – Приклад ініціуючого скрипта для випадку знаходження колонок, в яких був змінений тип даних

```
CREATE TABLE account(
    user_id serial PRIMARY KEY,
    username VARCHAR (50) UNIQUE NOT NULL,
    password VARCHAR (50) NOT NULL,
    email VARCHAR (355) UNIQUE NOT NULL,
    created_on TIMESTAMP NOT NULL,
    last_login TIMESTAMP
);
```

Рисунок 3.16 – Приклад поточної схеми БД для випадку знаходження колонок, в яких був змінений тип даних

Як можемо побачити з рисунків 3.15 та 3.16, в колонці last_login змінений тип даних last_login з TIMESTAMP до VARCHAR (355).

```
Some problems with database!!!
Problems with:
['account----last_login----TIMESTAMP'] ['account----last_login----VARCHAR(355)']

Founded datatype changing, let's restore their
You need to execute:
ALTER TABLE account ALTER COLUMN last_login TYPE TIMESTAMP;
```

Рисунок 3.17 – Результат роботи програми для випадку знаходження колонок, в яких був змінений тип даних

Як бачимо, наш застосунок виявив зміну типу бази даних, та запропонував змінити тип даних до початкового.

```
29 @staticmethod
30 def change_data_type(first_without_second, second_without_first):
31     for item in first_without_second:
32         table_name = item.split('----')[0]
33         column_name = item.split('----')[1]
34         data_type = item.split('----')[2]
35         for item_2 in second_without_first:
36             if table_name == item_2.split('----')[0] and column_name == item_2.split('----')[1]:
37                 print("Founded datatype changing, let's restore their")
38                 print("You need to execute:")
39                 print("ALTER TABLE " + table_name + " ALTER COLUMN " + column_name + " TYPE " + data_type + ";\n")
40                 Corrector.remove_entry_from_list(first_without_second,
41                                                 table_name + "----" + column_name + "----" + data_type)
42                 Corrector.remove_entry_from_list(second_without_first,
43                                                 table_name + "----" + column_name + "----" + item_2.split('----')[2])
44             else:
45                 return True, first_without_second, second_without_first
46         return False, first_without_second, second_without_first
47
```

Рисунок 3.18 – Програмна реалізація для випадку знаходження колонок, в яких був змінений тип даних

Роздивимось програмний код, що зображений на рисунку 3.18. Як можемо побачити, функція працює наступним чином: перебираються усі елементи, що не відповідають початковому скрипту, та якщо ім'я таблиці та ім'я колонки співпало, а тип даних не співпав, то пропонується варіант виправлення.

3.3.3 Знаходження видалених колонок

Інший варіант виправлення — знаходження видалених колонок. Для прикладу приведемо наступні скрипти:

```
CREATE TABLE account(
  user_id serial PRIMARY KEY,
  username VARCHAR (50) UNIQUE NOT NULL,
  password VARCHAR (50) NOT NULL,
  email VARCHAR (355) UNIQUE NOT NULL,
  created_on TIMESTAMP NOT NULL,
  last_login TIMESTAMP
);
```

Рисунок 3.19 – Приклад ініціуючого скрипта для випадку знаходження видалених колонок

```
CREATE TABLE account(
  user_id serial PRIMARY KEY,
  username VARCHAR (50) UNIQUE NOT NULL,
  password VARCHAR (50) NOT NULL,
  email VARCHAR (355) UNIQUE NOT NULL,|
  last_login TIMESTAMP
);
```

Рисунок 3.20 – Приклад поточної схеми БД для випадку знаходження видалених колонок

Як можемо побачити з рисунків 3.19 та 3.20, колонка `created_on` була видалена, тому виконаємо наш скрипт на даному випадку:

```
Some problems with database!!!
Problems with:
['account----created_on----TIMESTAMP'] []

Founded deleted columns, let's add them
You need to execute:
ALTER TABLE account ADD COLUMN created_on TIMESTAMP;
```

Рисунок 3.21 – Результат роботи програми для випадку знаходження видалених колонок

Програма пропонує додати видалену колонку командою

`ALTER TABLE account ADD COLUMN created_on TIMESTAMP;`

```
63 # Finding deleted columns
64 if len(first without second) > 0:
65     print("Founded deleted columns, let's add them")
66     print("You need to execute:")
67     for item in first without second:
68         splitted = item.split('----')
69         print("ALTER TABLE " + splitted[0] + " ADD COLUMN " + splitted[1] + " " + splitted[2] + " " + splitted[3] + ";\n")
70
```

Рисунок 3.22 – Програмна реалізація для випадку знаходження видалених колонок

Програмний код зображений на рисунку. Ідея полягає в тому, що, якщо є елементи, які є в початковому скрипті, та немає тих же елементів, що є в поточній схемі, то їх пропонується додати.

3.3.4 Знаходження зайвих колонок

Інший варіант виправлення — знаходження зайвих колонок. Для прикладу приведемо наступні скрипти:

```
CREATE TABLE account(
  user_id serial PRIMARY KEY,
  username VARCHAR (50) UNIQUE NOT NULL,
  password VARCHAR (50) NOT NULL,
  email VARCHAR (355) UNIQUE NOT NULL,
  created_on TIMESTAMP NOT NULL,
  last_login TIMESTAMP
);
```

Рисунок 3.23 – Приклад ініціуючого скрипта для випадку знаходження зайвих колонок

```
CREATE TABLE account(
  user_id serial PRIMARY KEY,
  username VARCHAR (50) UNIQUE NOT NULL,
  password VARCHAR (50) NOT NULL,
  email VARCHAR (355) UNIQUE NOT NULL,
  created_on TIMESTAMP NOT NULL,
  last_login TIMESTAMP,
  new_malicious_column VARCHAR (50) NOT NULL
);
```

Рисунок 3.24 – Приклад поточної схеми БД для випадку знаходження зайвих колонок

Як можемо побачити з рисунків 3.23 та 3.24, колонка `new_malicious_column` була додана до поточної схеми бази даних, тому виконаємо наш скрипт на даному випадку:

```
Some problems with database!!!
Problems with:
[] ['account----new_malicious_column----VARCHAR(50)']

Founded new columns, let's remove them
You need to execute:
ALTER TABLE account DROP new_malicious_column;
```

Рисунок 3.25 – Результат роботи програми для випадку знаходження зайвих колонок

Програма пропонує додати видалену колонку командою
`ALTER TABLE account DROP new_malicious_column;`

```
71 # Finding new columns
72 if len(second_without_first) > 0:
73     print("Founded new columns, let's remove them")
74     print("You need to execute:")
75     for item in second_without_first:
76         splitted = item.split('----')
77         print("ALTER TABLE " + splitted[0] + " DROP " + splitted[1] + ";\n")
78
```

Рисунок 3.26 – Програмна реалізація для випадку знаходження зайвих колонок

Програмний код зображений на рисунку та дуже схожий з програмним кодом 3.3.3. Ідея полягає в тому, що, якщо є елементи, які є в поточному скрипті, та немає тих же елементів, що є в початковій схемі, то їх пропонується видалити.

3.3.5 Комбінація випадків

Також, програма знаходить комбінацію випадків. Роздивимось наступні скрипти:

```
CREATE TABLE account(
  user_id serial PRIMARY KEY,
  password VARCHAR (50) NOT NULL,
  email VARCHAR (355) UNIQUE NOT NULL,
  created_on TIMESTAMP NOT NULL,
  last_login TIMESTAMP
);
```

Рисунок 3.27 – Приклад ініціюючого скрипта для комбінації випадків

```
CREATE TABLE account(
  user_id serial PRIMARY KEY,
  username VARCHAR (50) UNIQUE NOT NULL,
  password VARCHAR (50) NOT NULL,
  email VARCHAR (355) UNIQUE NOT NULL,
  created_on TIMESTAMP NOT NULL,
  new_malicious_column VARCHAR (50) NOT NULL
);

CREATE TABLE fake_table(
  fake_id serial PRIMARY KEY,
  fake_column_1 VARCHAR (50) UNIQUE NOT NULL,
  fake_column_2 VARCHAR (50) NOT NULL
);
```

Рисунок 3.28 – Приклад поточної схеми БД для комбінації випадків

Як можемо побачити з рисунків 3.27 та 3.28, в поточній схемі існує нова таблиця `fake_table` також існує дві нові колонки, а саме `username VARCHAR(50)` та `new_malicious_column VARCHAR(50)` та видалена колонка `last_login TIMESTAMP`. Запуск нашої програми видав наступні результати:

```
Some problems with database!!!
Problems with:
['account----last_login----TIMESTAMP'] ['account----new_malicious_column----VARC
HAR(50)', 'account----username----VARCHAR(50)', 'fake_table----fake_column_1----
VARCHAR(50)', 'fake_table----fake_column_2----VARCHAR(50)']

Founded new table, let's remove their
You need to execute:
DROP TABLE fake_table;

Founded deleted columns, let's add them
You need to execute:
ALTER TABLE account ADD COLUMN last_login TIMESTAMP;

Founded new columns, let's remove them
You need to execute:
ALTER TABLE account DROP new_malicious_column;

ALTER TABLE account DROP username;
```

Рисунок 3.29 – Результат роботи програми для комбінації випадків

Як можемо побачити, він виявив усі зміни схеми бази даних, та запропонував наступні SQL команди:

```
DROP TABLE fake_table;

ALTER TABLE account ADD COLUMN last_login TIMESTAMP;

ALTER TABLE account DROP new_malicious_column;

ALTER TABLE account DROP username;
```

Усі ці команди приведуть поточну схему бази даних до початкової.

Висновки до розділу 3

У цьому розділі була описана методика виявлення та виправлення змін схеми бази даних, а саме описана програма класів, діаграма високого рівня та діаграма послідовностей. Також, був представлений програмний код кожного з модулів застосунку та додатково, для кожного з чотирьох варіантів змін схеми баз даних було детально описано функції, які відповідають за виправлення та представлення результати, а саме вивід програмного скрипта.

Додатково було продемонстровано складний випадок з комбінацією усіх порушень схеми бази даних, та команди які програма запропонувала ввести для приведення схеми бази даних.

Для подальшого розвитку програми, можливо доповнювати модуль виправлення додатковим функціоналом та винятковими випадками, але, згідно до магістерської дисертації, можна сказати, що запропонована методика має науковий сенс та може бути запроваджена як реальний проект.

4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї стартап-проекту

На першому етапі слід проаналізувати зміст ідеї, можливі напрямки застосування, основні вигоди, що може отримати користувач товару (Таблиця 4.1) і чим відрізняється від існуючих аналогів та замінників (Таблиця 4.2). В останньому випадку - визначення переліку слабких, сильних та нейтральних характеристик та властивостей ідеї потенційної послуги є підґрунтям для формування її конкурентоспроможності.

Таблиця 4.1 – Опис ідеї стартап-проекту

<i>Зміст ідеї</i>	<i>Напрямки застосування</i>	<i>Вигоди для користувача</i>
Створення продукту, який буде знаходити та виправляти помилки схеми бази даних.	1. Компанії середнього і великого бізнесу.	1. Експрес-діагностика. Виявлення помилок в впродовж декількох секунд. Зручне інформаційне повідомлення підкаже фахівцеві, для яких об'єктів необхідно провести виправлення.
	2. Середні та великі проекти в ІТ-компаніях.	2. Економія ресурсів. Швидкість перевірки схеми бази даних мінімізує витрачений час і інші ресурси.

Кінець таблиці 4.1

	3. Банки, які мають за основу контролювати базу даних, щоб не допустити вторгнення у неї та “злиття” бази даних клієнтів.	3. Підтвердження цілісності і надійності. Ви будете точно знати, що база даних робочого середовища містить всі необхідні системою таблиці / поля.
	4. Організації, що користуються системою 1С, а саме: бухгалтерські відділи, фінансові установи і тд.	4. Можливість контролювати вторгнення до бази даних

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї стартап-проекту

№ п / п	Техніко - економічні характери стики ідеї	(потенційні) товари/концепції конкурентів			W	N	S
		Мій проект	Database structure check	DLM Dashboar d			
1 .	Швидкість роботи	константний час	лінійна залежність від розміру	працює в фоні	Працює довго, залежить від факторів	-	Працює швидко, не залежить від факторів

Продовження таблиці 4.2

2	Можливість виправлення	Є можливість виправлення змін в схемі бази даних	Наявність можливості провести повторне оновлення схеми бази даних.	Немає можливості виправлення	Необхідність наявності технічного спеціаліста	-	Можливість виправлення та повернення до початкового стану бази даних.
3	Знаходження змін схеми бази даних	Є можливість знаходження змін схеми бази даних	Є можливість знаходження змін схеми бази даних	Є можливість знаходження змін схеми бази даних	БД погано захищене	Затрати певного проміжку часу	Можливість діагностики на несанкціоновані зміни

Кінець таблиці 4.2

4	Залежність від розміру бази даних	Не залежить від розміру бд	Не залежить від розміру бд	Залежить від розміру бд	Програма довго працює, в залежності від розміру бази даних	Програма працює довше, в залежності від розміру бази даних, але в межах норми	Час роботи не залежить від розміру бази даних
5	Виявлення цілісності об'єктів бази даних	Існує виявлення	Існує виявлення	Існує виявлення	Не виявляє цілісності об'єктів бази даних	Можливі некоректні виявлення	Виявляє цілісність об'єктів бази даних
6	Необхідність в резервних копіях	Немає необхідності в резервних копіях	Невідомо	Є необхідності в резервних копіях	Факт необхідності резервних копій	-	Можливість повернутися до попередньої версії

4.2 Технологічний аудит ідеї проекту

Далі необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення послуги) (Таблиця 4.3).

Таблиця 4.3 - Технологічна здійсненність ідеї проекту

<i>№ n/n</i>	<i>Ідея проекту</i>	<i>Технології її реалізації</i>	<i>Наявність технологій</i>	<i>Доступність технологій</i>
1	Аналіз скриптів ініціалізації бази даних	Використання скриптів ініціалізації бази даних за допомогою встроєних функцій бази даних	Наявний ручний спосіб вилучення ініціюючого скрипта бази даних	Технологія є доступною
2	Аналіз поточного стану схеми бази даних	Використання програм або команд для аналізу поточного стану схеми бази даних	Наявні команди, що реалізують дану ідею, а також програми	Технології є доступними і мають графічний інтерфейс, деякі – консольні застосунки.

Кінець таблиці 4.3

3	Використання методу приведення *.sql файлів до вигляду формату *.json	Використання Python3 та встроєної бібліотеки JSON	Технології є наявними	Технології є доступними
4	Використання методу порівняння схем бази даних	Використання алгоритму порівняння	Наявні деякі алгоритми порівняння файлів з даними	Технології є доступними
5	Використання технології виправлення змін у бд	Використання алгоритмів, існуючих методів виправлення змін у бд	Наявні існуючі технології такі як: ручний метод виправлення, відновлення з резервних копій, а також розроблений мною підхід	Технології є доступними
Обрана технологія реалізації ідеї проекту: так як для реалізації ідеї проекту, всі технології є наявними та доступними, тому обираються всі вище описані технології.				

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Для врахування стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів проаналізуємо наявність попиту, обсяг, динаміку розвитку ринку (Таблиця 4.4 - 4.5). За результатами аналізу бачимо, що за попереднім оцінюванням ринок привабливий для входження.

Таблиця 4.4 - Характеристика потенційного ринку стартап-проекту

<i>№ n/n</i>	<i>Показники стану ринку (найменування)</i>	<i>Характеристика</i>
1	Кількість головних гравців, од	10
2	Загальний обсяг продаж, грн/ум.од	1 млн ум.од.
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Потреба у кадрах із високим рівнем компетентності у сфері баз даних
5	Специфічні вимоги до стандартизації та сертифікації	Не потребує сертифікації та стандартизації
6	Середня норма рентабельності в галузі (або по ринку), %	Не менше 150

На основі проведеного дослідження є можливість стверджувати про привабливість проекту для входження на ринок за попереднім оцінюванням.

Таблиця 4.5 - Характеристика потенційних клієнтів стартап-проекту

<i>№ п/ п</i>	<i>Потреба, що формує ринок</i>	<i>Цільова аудиторія (цільові сегменти ринку)</i>	<i>Відмінності у поведінці різних потенційних цільових груп клієнтів</i>	<i>Вимоги споживачів до товару</i>
	Виявлення та виправлення порушень цілісності схеми бази даних, що можуть призвести до втрат даних або відмові в обслуговуванні бази даних	<p>1. Компанії середнього і великого бізнесу.</p> <p>2. Середні та великі проекти в ІТ-компаніях.</p> <p>3. Банки, які мають за основу контролювати базу даних, щоб не допустити вторгнення у неї та “злиття” бази даних клієнтів.</p> <p>4. Організації, що користуються системою 1С, а саме: бухгалтерські відділи, фінансові установи і тд.</p>	<p>Для кожної з категорій існують окремі цінності пропозицій.</p> <p>Пропозиція відрізняється для кожної цільової групи.</p>	<p>- виявлення порушень цілісності схеми бд</p> <p>- виправлення порушень цілісності схеми бд</p> <p>- приведення бд до початкового стану</p> <p>- швидкість роботи</p>

Далі проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (Таблиця №4.6-4.7). Фактори в таблиці подано в порядку зменшення значущості. Також проведено визначення загальних рис конкуренції на ринку (Таблиця 4.8).

Таблиця 4.6 - Фактори загроз

<i>№ n/n</i>	<i>Фактор</i>	<i>Зміст загрози</i>	<i>Можлива реакція компанії</i>
1	Цінова конкуренція	Наявні деякі продукти у вільному доступі	Покращення роботи власного продукту відносно конкурентів
2	Низька точність виявлення зміни схеми бази даних	Не всі зміни схеми бази даних можуть бути виявленні	Покращення продукту, шляхом вивчення специфічних ситуацій
3	Зниження доходів потенційних споживачів	Зниження купівельної спроможності клієнтів	Вимушене зменшення обсягів виробництва
4	Не точні вказівки для виправлення	Вказівки для виправлення не призводять до відновлення схеми бази даних	Покращення алгоритму виправлення, запровадження нових підходів, вивчення специфічних ситуацій

Таблиця 4.7 - Фактори можливостей

<i>№ n/n</i>	<i>Фактор</i>	<i>Зміст можливості</i>	<i>Можлива реакція компанії</i>
1	Поява нового методу представлення початкового скрипту	Використання іншої репрезентації початкового скрипта, що дозволить зберегти більше зв'язаної інформації	Підвищення результатів виявлення змін схеми бази даних, можливість виявляти інші атаки на бази даних
2	Поява нового методу виправлення змін схеми бази даних	Використання іншого алгоритму для порівняння та виправлення змін бази даних	Можливість виправляти інші атаки на бази даних, обробка специфічних ситуацій
3	Поява нового типу бази даних з строгою мовою програмування та сталою схемою	Новий тип баз даних, який буде використовуватися та мати переваги над реляційною базою даною та матиме строгу схему та мову програмування	Запровадження даного підходу на інший тип бази даних, що поширить сферу використання методу

Кінець таблиці 4.7

4	Поява нового підходу для отримання поточної схеми бази даних	Використання стороннього програмного продукту, який буде отримувати скрипт, що буде більше схожий на початковий скрипт	Впровадження додаткових алгоритмів, що будуть виявляти та виправляти додаткові зміни бази даних
5	Запровадження глибинного навчання для виправлення змін схеми бази даних	Використання деякої топології нейронної мережі, що зможе привести схему бази даних до початкового	Впровадження глибинного навчання, для генерації вказівок виправлення схеми бази даних

Таблиця 4.8 - Ступеневий аналіз конкуренції на ринку

<i>Особливості конкурентного середовища</i>	<i>В чому проявляється дана характеристика</i>	<i>Вплив на діяльність підприємства</i>
Олігополістична конкуренція	На ринку присутні продукти, які схожі за принципом дії та доступні у вільному доступі	Збільшення інформованості потенційних клієнтів про якість послуг, що надаються у продукті та про наявність інноваційних методів виявлення та вирішення змін у схемі бази даних, що може привабити потенційних клієнтів
За рівнем конкурентної боротьби - національний	Надання послуг для різних груп та типів клієнтів в Україні та за її межами	Застосування нових технологій та підходів для вдосконалення рішення
3. За галузевою ознакою - внутрішньогалузев а	Усі методики та технології, що застосовуються у аналізах, є вузькоспеціалізованими та вузько направленними	Необхідно добре зарекомендувати продукт та зосередити зусилля на пошуку конкурентних переваг, які дозволять компанії займати стійкі конкурентні позиції

Кінець таблиці 4.8

4. Конкуренція за видами товарів - товарно-родова	Рішення, що використовується для задоволення потреб клієнтів, але істотно відрізняються від рішень конкурентів.	Надання послуг з виявлення та виправлення змін схеми бази даних на основі глибинного навчання
5. За характером конкурентних переваг - цінова	Ціна запропонованого рішення є меншою за рахунок використання вже готових доступних результатів наукових досліджень та розробок.	Поєднання бюджетності та доступності з підтриманням високого рівня якості надання послуг
6. За інтенсивністю - марочна	Сукупність характеристик та властивостей рішення	Підвищення якості роботи програмного рішення для клієнтів

Більш детальний аналіз умов конкуренції в галузі можна провести за моделлю сил М. Портера (табл. 4.9).

Таблиця 4.9 - Аналіз конкуренції в галузі за М. Портером

<i>Складові аналізу</i>	<i>Прямі конкуренти в галузі</i>	<i>Потенційні конкуренти</i>	<i>Постачальники</i>	<i>Клієнти</i>	<i>Товари - заміники</i>
	відсутні	Database structure check for Creatio, DLM Dashboard	Досконалі продукти, підтримка зі сторони розробки, рішення перевірене часом	Великі ІТ компанії, середні підприємства та банки мають великі фінанси та підтримують високу якість продуктів, що їх необхідні для хорошого функціонування	Товари і заміники відсутні
Висновки:	Продукт є дослідницьким, тому не є конкурентом	Є можливість входу на ринок, але існують конкуренти, що вже працюють на цьому ринку.	Постачальники диктують умови роботи на ринку; компанії, які мають найбільше ресурсів, мають найбільш якісний продукт.	Клієнти диктують умови на ринку; при організації вибору послуги, відчутний рівень відношення до вартості рішення та його якості.	Обмеження на ринку через товари заміники немає.

З огляду на конкурентну ситуацію, можливість роботи на ринку існує. Щоб бути конкурентоспроможним, продукт повинен мати такі сильні сторони, як доступність, висока якість послуг, що надаються, розповсюдженість інформації про дану марку. У (Таблиця 4.10) визначено та обгрунтовано перелік факторів конкурентоспроможності.

Таблиця 4.10 - Обгрунтування факторів конкурентоспроможності

<i>№ n/n</i>	<i>Фактор конкурентоспроможності</i>	<i>Обгрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)</i>
1	Ціна	Чим вигіднішою є ціна для споживача, тим вірогідніше його вибір.
2	Якість продукту	Продукт працює швидко та показує гарні показники виявлення та виправлення
3	Доступ до ресурсів у конкурентів	Так як потенційні конкуренти вже працюють на ринку, вони мають клієнтів та встановлений шлях продажів та маркетингу, тому вони мають більше ресурсів як матеріальних, так і інформаційних
4	Асортимент	В умовах збільшення інтенсивності між існуючими конкурентами завоювання споживачів відбувається за рахунок нових методів та алгоритмів.

За визначеними у (Таблиця 4.10) факторами конкурентоспроможності проводимо аналіз сильних та слабких сторін стартап-проекту (Таблиця 4.11).

Таблиця 4.11 - Порівняльний аналіз сильних та слабких сторін стартап-проекту

№ n/n	Фактор конкурентоспроможност <i>i</i>	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з стартап-проектом						
			-3	-2	-1	0	+1	+2	+3
1	Ціна	15					+		
2	Якість продукту	20			+				
3	Доступ до ресурсів у конкурентів	10							+
4	Асортимент	10		+					

На основі ринкових загроз та можливостей, та сильних і слабких сторін, виділених у попередній таблиці, складаємо SWOT-аналіз. Це матриці аналізу сильних та слабких сторін, загроз та можливостей (Strength, weak, troubles, opportunities відповідно). Це фінальний етап ринкового аналізу можливостей впровадження проекту (Таблиця 4.12)

Таблиця 4.12 - SWOT - аналіз стартап-проекту

<p>Сильні сторони:</p> <p>Не потребує ресурсів сервера, для зберігання та обробки резервних копій, підхід простий у реалізації, тому низька ціна реалізації</p>	<p>Слабкі сторони:</p> <p>Низький рівень маркетингу, дуже обмежена кількість правил для виявлення та виправлення</p>
<p>Можливості:</p> <p>залучення компаній, що працюють з базами даних, які мають доступ до веб-мережі, в якості постачальників чи клієнтів</p>	<p>Загрози:</p> <p>конкуренція відносно великих компаній, з великими фінансовими та розумовими активами</p>

Далі розробляємо альтернативи ринкової поведінки (Таблиця 4.13) для виведення стартап-проекту на ринок та орієнтовний оптимальний час ринкової реалізації з огляду на потенційні проекти конкурентів. Альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів.

Таблиця 4.13 - Альтернативи ринкового впровадження стартап-проекту

<i>№ п/п</i>	<i>Альтернатива (орієнтовний комплекс заходів) ринкової поведінки</i>	<i>Ймовірність отримання ресурсів</i>	<i>Строки реалізації</i>
1	Створення угоди з певною ІТ компанією	Висока ймовірність отримання ресурсів	До 1 року
2	Впровадження рекламних засобів для таргетингової категорії	Середня ймовірність отримання ресурсів	До 6 місяців
3	Участь в ІТ виставках, публікація в журналах	Середня ймовірність отримання ресурсів	До 1 року
4	Збільшення представленості	Середня ймовірність отримання ресурсів	До 1 року

4.4 Розроблення ринкової стратегії проекту

Для розроблення ринкової стратегії в першу чергу необхідним є визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (Таблиця 4.14).

Таблиця 4.14 - Вибір цільових груп потенційних споживачів

<i>№ п/п</i>	<i>Опис профілю цільової групи потенційн их клієнтів</i>	<i>Готовність споживачів сприйняти продукт</i>	<i>Орієнтов ний попит в межах цільової групи</i>	<i>Інтенсивність конкуренції в сегменті</i>	<i>Просто та входу у сегмент</i>
1	Середні та великі ІТ компанії	Клієнти потребують продукт такого типу та готові ним користуватись	Високий рівень попиту	Існують рішення зі схожим функціоналом, однак які не є заміниками	Простий
2	Банки	Клієнти потребують продукт такого типу та готові ним користуватись	Високий рівень попиту	Існують рішення зі схожим функціоналом, однак які не є заміниками	Простий

Кінець таблиці 4.14

3	Організації, що користуються системою 1С	Клієнти потребують продукт такого типу та готові ним користуватись	Середній рівень попиту	Існує конкуренція з подібними функціями, проте мінімальна	Простий
Які цільові групи обрано: середні та великі ІТ компанії та банки, адже вони мають високий рівень попиту.					

Отже, на основі аналізу цільових груп потенційних споживачів було виявлено, що існує високий попит та невелика конкуренція на наступні цільові групи: середні та великі ІТ компанії та банки, тому вони і створюють цільову групу.

За результатами аналізу потенційних груп споживачів (сегментів) обрано стратегію диференційованого маркетингу.

Таблиця 4.15 - Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Флангова атака	Стратегія диференційованого маркетингу	Сильні сторони та можливості рішення	Стратегія диференціації

Наступним кроком є вибір стратегії конкурентної поведінки (табл. 4.16).

Таблиця 4.16 - Визначення базової стратегії конкурентної поведінки

<i>№ п/ п</i>	<i>Чи є проект «першопрохідцем» на ринку?</i>	<i>Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?</i>	<i>Чи буде компанія копіювати основні характеристики товару конкурента, і які?</i>	<i>Стратегія конкурентної поведінки*</i>
1	Проект є варіантом, який пропонує нову методику, проте на ринку вже є інші готові методики, що можуть стати конкурентами	Так	Частково	Стратегія слідування лідеру

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту, а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки розробляється стратегія позиціонування, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.17 - Визначення стратегії позиціонування

<i>№ п/п</i>	<i>Вимоги до товару цільової аудиторії</i>	<i>Базова стратегія розвитку</i>	<i>Ключові конкурентоспроможні позиції власного стартап-проекту</i>	<i>Вибір асоціацій, які мають сформувану комплексну позицію власного проекту (три ключових)</i>
1	Продукт, швидкість та точність роботи якого переважає над конкурентами, зі зручним користувацьким інтерфейсом, ціна якого не перебільшує ціну конкурентів	Стратегія диференціації	Не потребує ресурсів сервера, для зберігання та обробки резервних копій, підхід простий у реалізації, тому низька ціна реалізації	Швидкість, точність виявлення, можливість виправлення, нижча цінова політика

4.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування *маркетингової концепції товару*, який отримає споживач. Для цього у табл. 4.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18 - Визначення ключових переваг концепції потенційного товару

<i>№ n/n</i>	<i>Потреба</i>	<i>Вигода, яку пропонує товар</i>	<i>Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)</i>
1	Знаходження можливих змін схеми бази даних	Висока точність виявлення змін бази даних	Унікальний продукт який виявляє зміни схеми бази даних
2	Можливість виправлення схеми бази даних	Висока точність вказівок для виправлення схеми бази даних	Продукт, який може запропонувати варіанти виправлення, тобто приведення схеми до початкового стану
3	Зручний користуваць кий інтерфейс	Наявність легкого запуску програми	Легкий запуск додатку

Таблиця 4.19 - Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Система виявлення та виправлення цілісності схеми бази даних на основі скриптів ініціалізації.		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Використовує початкові скрипти для перевірки схеми	М	Тх
	2. Швидко працює	М	Тх
	3. Має простий інтерфейс	М	Тх
	4. Може запропонувати варіант вирішення змін	М	Тх
	Якість: виявляються зміни схеми бази даних		
	Пакування: Надання користувачу електронної ліцензії (ключа доступу)		
Марка: D&R			
III. Товар із підкріпленням	До продажу: початок рекламної компанії, пропонування демо-версії		
	Після продажу: продовження рекламної компанії		
За рахунок чого потенційний товар буде захищено від копіювання: користувач не має початкового коду, а має лише виконувані файли			

Таблиця 4.20 - Визначення меж встановлення ціни

<i>№ п/п</i>	<i>Рівень цін на товари- замінники</i>	<i>Рівень цін на товари- аналоги</i>	<i>Рівень доходів цільової групи споживачів</i>	<i>Верхня та нижня межі встановлення ціни на товар/послугу</i>
	2000	800	1200	900-1000

Таблиця 4.21 - Формування системи збуту

<i>№ п/п</i>	<i>Специфіка закупівельної поведінки цільових клієнтів</i>	<i>Функції збуту, які має виконувати постачальник товару</i>	<i>Глибина каналу збуту</i>	<i>Оптимальна система збуту</i>
	Купівля напряму	Організувати широку мережу збуту товару та підтримку	1	Пряма

Останньою складовою маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 4.22).

Таблиця 4.22 - Концепція маркетингових комунікацій

<i>№ п/ п</i>	<i>Специфіка поведінки цільових клієнтів</i>	<i>Канали комунікацій , якими користуют ься цільові клієнти</i>	<i>Ключові позиції, обрані для позиціонува ння</i>	<i>Завдання рекламного повідомленн я</i>	<i>Концепція рекламного звернення</i>
1	Дослідженн я переваг та властивосте й продукту для задоволення своїх потреб	Інформацій ні канали	Оцінювання цілісності схеми бази даних від зловмисних діянь	Донести переваги до потенційних клієнтів	Реклама, що демонструє якість товару, його цінність або можливості експлуатації

Висновки до розділу 4

Після проведення даного маркетингового аналізу можна зробити висновок, що є висока можливість ринкової комерціалізації проекту, оскільки наявний високий попит, динаміка ринку зростаюча, рентабельність роботи на ринку є задовільною. До сильних сторін проекту можна віднести те, що він простий у реалізації, тому низька ціна реалізації. Також перевагою проекту є його наукова місткість і новизна, та можливість залучення компаній, що працюють з базами даних, які мають доступ до веб-мережі, в якості постачальників чи клієнтів. Не зважаючи на існуючі незначні бар'єри входження та стан конкуренції, впровадження з огляду на потенційні групи клієнтів і високу конкурентоспроможність проекту є високopersпективним. Я вважаю, що є доцільною подальша робота над реалізацією даного проекту.

ВИСНОВКИ

У даній дипломній роботі представлена методика виявлення та виправлення порушень цілісності схеми бази даних на основі скриптів ініціалізації. Проблема цілісності структури бази даних, тобто, зміна схеми бази даних відбувається внаслідок несанкціонованих змін, одним з прикладів, можуть бути SQL injection атаки.

Впродовж роботи було розглянуто та проаналізовано типи баз даних, існуючі загрози на бази даних за даними OWASP, code injection атаки і особливу увагу було приділено SQL injection атакам, їх наслідкам для реляційних баз даних та існуючим методам захисту від загроз на базу даних.

У ході дослідження також було проаналізовано існуючі методи виявлення та виправлення порушень бази даних, та основуючись на їх недоліках запропоновано нову методику. Дана методика складається з наступних етапів: етап представлення та аналізування початкових скриптів, етап аналізування поточного стану схеми бази даних, етап порівняння схем, етап реагування та виправлення, результатом якого буде приведення схеми до початкової схеми. Програмна реалізація методики була реалізована на мові програмування Python3.

Розроблена методика є повним рішенням, що дозволяє не тільки виявляти порушення схеми бази даних, а й пропонувати варіанти вирішення, чого не існує в науковому співтоваристві. Дана методика використовує скрипти ініціалізації схеми бази даних для порівняння з поточною схемою, чого також не описано та не реалізовано.

Запропонована методика може бути використана для реалізації корпоративного продукту або продукту з відкритим початковим кодом, який зможе бути корисним як провідним компаніям в ІТ сфері, так і невеликим компаніям, які використовують бази даних в бізнесі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- 1 Что такое база данных. Система управления базами данных [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: <https://hostiq.ua/wiki/database/>.
- 2 Мирошниченко Е. А. К формальному определению понятия «база данных» [Текст] // Пробл. информатики. – М., 2011. – Разд. 2. – С. 83 – 87.
- 3 ГОСТ ИСО МЭК ТО 10032-2007. Эталонная модель управления данными[Текст]. – На заміну ISO/IEC TR 10032 - 2003 Information technology — Reference model of data management
- 4 Types of databases [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: <https://searchsqlserver.techtarget.com/definition/database>.
- 5 Bernard Marr in blog titled as Ten top languages for crunching Big Data on Data Science Central. The online resource for Big Data Practisioners [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: <https://www.datasciencecentral.com/profiles/blogs/ten-top-languages-for-crunching-bigdata>.
- 6 Dinesh Thakur in his website named ECOMPUTER NOTES in chapter titled as “What are Relational Algebra and Relational Calculus” [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: <http://ecomputernotes.com/database-system/rdbms/relational-algebra-and-relationalcalculus>.
- 7 Relational algebra [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: https://en.wikipedia.org/wiki/Relational_algebra.
- 8 Browser Architecture and Mobile Applications [Электронный ресурс]. — 2017. — Режим доступа до ресурсу: <https://seng130.wordpress.com/lectures-2/web-servers/>.
- 9 Multitier architecture [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: https://en.wikipedia.org/wiki/Multitier_architecture.
- 10 Top 10 Database Attacks [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: <https://www.bcs.org/content-hub/top-ten-database-attacks>.

- 11 Code injection [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: https://en.wikipedia.org/wiki/Code_injection.
- 12 The Cross-site Scripting (XSS) Vulnerability: Definition and Prevention [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: <https://www.netsparker.com/blog/web-security/cross-site-scripting-xss/>.
- 13 Rain forest Puppy. NT Web Technology Vulnerabilities [Текст] / Rain forest Puppy// Phrack Magazine. — 1998. — Volume 8. — Issue 54.
- 14 Naive algorithm for Pattern Searching [Электронный ресурс]. — 2018. — Режим доступа до ресурсу: <http://www.geeksforgeeks.org/searching-for-patterns-set-1-naive-pattern-searching/>.
- 15 SQL injection [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: https://en.wikipedia.org/wiki/SQL_injection.
- 16 Почему SQL Injection - это самый опасный вид уязвимостей? [Электронный ресурс]. — 2018. — Режим доступа до ресурсу: https://acribia.ru/articles/why_sql_injection_is_the_most_dangerous_type_of_vulnerability.
- 17 Stephen Kost. AN INTRODUCTION TO SQL INJECTION ATTACKS FOR ORACLE DEVELOPERS [Текст] / Stephen Kost . – М.: Integrity Corporation, 2007 . – 50с.
- 18 Защита от SQL инъекций [Электронный ресурс]. — 2012. — Режим доступа до ресурсу: <http://www.securityscripts.ru/articles/sql-injection.html>.
- 19 Как защититься от SQL инъекций [Электронный ресурс]. — 2018. — Режим доступа до ресурсу: <http://lifeexample.ru/php-primeryi-skriptov/zashhita-ot-sql-inektsiy.html>.
- 20 DBCC CHECKDB (Transact-SQL) [Электронный ресурс]. — 2017. — Режим доступа до ресурсу: <https://docs.microsoft.com/en-us/sql/t-sql/database-console-commands/dbcc-checkdb-transact-sql?view=sql-server-ver15#checking-objects-in-parallel>.
- 21 Логирование в MySQL [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: <https://ruhighload.com/Логирование+в+mysql>.

- 22 The DDL Log [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: <https://dev.mysql.com/doc/refman/5.7/en/ddl-log.html>.
- 23 Database Recovery Techniques in DBMS [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: <https://www.geeksforgeeks.org/database-recovery-techniques-in-dbms/>.
- 24 SQL [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: <https://uk.wikipedia.org/wiki/SQL>.
- 25 Creating a Table [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: <https://dev.mysql.com/doc/refman/8.0/en/creating-tables.html>.
- 26 Exporting MySQL, PostgreSQL and SQL Server schema structure [Электронный ресурс]. — 2019. — Режим доступа до ресурсу: <https://www.eversql.com/exporting-mysql-schema-structure-to-xml-using-mysql-clients/#mysqldump>.