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

«До захисту допущено»
В.о. завідувача кафедри

_____ М.В.Грайворонський
(підпис)
“___”_____2019 р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.170101 «Безпека інформаційних і комунікаційних систем»

на тему: Моделювання систем багаторівневої безпеки баз даних_____

_____

_____

Виконав (-ла): студент (-ка) _4__ курсу, групи __ФБ-52_____
(шифр групи)

_____Головач Тетяна _____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник К.т.н., доцент Коломицев М.В._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

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

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2019 року

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

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.170101 «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ М.В.Грайворонський
(підпис)
«___»_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту

_____ Головач Тетяна _____
(прізвище, ім'я, по батькові)

1. Тема роботи Моделювання систем багаторівневої безпеки баз даних _____
_____
_____ ,
науковий керівник роботи К.т.н., доцент Коломицев М.В. _____
_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «____ » 2019 р. № _____

2. Термін подання студентом роботи 10 червня 2019 р.

3. Вихідні дані до роботи _____
_____
_____

4. Зміст роботи_____
_____
_____
_____
_____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) _____
_____
_____
_____

РЕФЕРАТ

Кваліфікаційна робота: 81 сторінка, 19 рисунків, 5 таблиць, 1 додаток, 16 джерел.

Об’єкт дослідження: моделі багаторівневої безпеки баз даних.

Предмет досліджень: впровадження моделей багаторівневої безпеки БД(SeaView, Sandhu–Jajodia, Smith–Winslett, Multilevel Relational) у сучасні системи та продуктивність отриманих систем.

Мета роботи: створення та опис методик моделювання різних підходів до багаторівневої безпеки баз даних, що дозволяють аналізувати властивості цих підходів.

Методи дослідження: емпіричний, а саме аналіз, моделювання, експеримент та порівняння.

При написанні роботи був проведений аналіз і узагальнення наукової літератури, була змодельована база даних із використанням різних підходів до багаторівневої безпеки та проведений експеримент, що дозволив порівняти та проаналізувати можливість використання даних моделей в сучасних системах.

Результатом роботи є розроблені алгоритми реалізації операцій реляційної алгебри в системах з багаторівневою безпекою і побудований на їх основі програмний комплекс моделювання баз даних з багаторівневою безпекою.

Ключові слова: конфіденційна інформація, управління доступом, безпека баз даних, багаторівнева безпека, системи багаторівневої безпеки баз даних.

ABSTRACT

Qualifying work: 81 pages, 19 pictures, 5 tables, 1 addition, 16 sources.

The object of research: models of multilevel database security.

The subject of research: introduction of the models of multilevel security for databases (SeaView, Sandhu-Jajodia, Smith-Winslett, Multilevel Relational) in modern systems and productivity of the systems received.

The purpose of the work is to create and describe a methodology for modeling different approaches to multilevel security, which allows analyzing the security properties of these units.

Research methods: empirical, namely analysis, modeling, experimentation and comparison.

For writing the work was conducted the analysis and development of scientific literature, which was used on the basis of data using different approaches to multilevel security and conduct an experiment that will allow comparing and analyzing the possibilities of using these models in modern systems.

The result of the work is developed algorithms for the implementation of relational algebra operations in a system with multilevel security and built on their basis software complex modeling databases with multilevel security.

Keywords: confidential information, access control, database security, multilevel security, multilevel security systems for databases.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – бази даних;

СУБД – системи управлінням бази даних;

RDB (Relational Databases) – реляційні бази даних;

RDBMS (Relational Databases Management System) – система управління реляційними базами даних;

SQL (Structured Query Language) - мова структурованих запитів;

DBA (Database Administrator) – адміністратор баз даних;

DAC (Discretory Access Control) – дискреційне управління доступом;

MAC (Mandatory Access Control) – мандатне управління доступом;

RBAC (Role based Access Control) – рольове управління доступом;

MLS (Multilevel Security) – багаторівнева безпека;

MLR (Multilevel Relational) – багаторівнева реляційна (модель);

TC(tuple classifier) - класифікатор запису – рівень безпеки запису.

ВСТУП

На сьогоднішній день існує безліч банків, компаній і корпорацій, що мають свої БД, а більшість державних установ має складні реєстраційно-облікові системи та продовжує активно працювати над створенням електронних сховищ даних, аналітичних та звітних систем, тому кількість баз даних, що містять конфіденційну інформацію, постійно зростає. Потреба у захисті БД є завжди, незалежно від того, це база з персональними даними клієнтів інтернет-магазину чи база даних банку. В більшості ситуацій належний захист конфіденційної інформації в базах даних відсутній, адже для цього потрібен підхід, що дозволяє більш гнучке розмежування доступу для працівників різних професій. Саме тому потрібно впроваджувати системи багаторівневої безпеки баз даних.

Безпечне керування конфіденційними даними є критичним питанням у сучасних організаціях. Багаторівневі системи баз даних пропонують один підхід до безпеки, який передбачає наявність різних рівнів безпеки даних в базі. Метою таких систем є обмін даними, що мають різні рівні безпеки, і запобігання несанкціонованому доступу до них. Моделі з багаторівневою безпекою зберігають інформацію на різних рівнях безпеки окремо для запобігання несанкціонованому доступу до даних користувачами на різних рівнях. В даній роботі розглядаються моделі багаторівневої безпеки баз даних та буде проаналізовано, чи можливе впровадження таких моделей в сучасні системи, що використовуються державними установами України.

Мета дослідження: створення та опис методик моделювання різних підходів до багаторівневої безпеки баз даних, що дозволяють аналізувати властивості цих підходів.

Були поставлені такі **завдання**:

- розглянути та проаналізувати концепцію багаторівневої безпеки баз даних;
- змоделювати системи багаторівневої безпеки БД;
- провести дослідження, в ході якого порівняти між собою моделі SeaView, Sandhu–Jajodia, Smith–Winslett, Multilevel Relational і проаналізувати можливість їх використання в сучасних системах.

Об’єкт дослідження: моделі багаторівневої безпеки БД.

Предмет дослідження: впровадження моделей багаторівневої безпеки БД(SeaView, Sandhu–Jajodia, Smith–Winslett, Multilevel Relational) у сучасні системи та продуктивність отриманих систем.

1 КОНЦЕПЦІЯ БЕЗПЕКИ БАЗ ДАНИХ

1.1 Основні поняття інформаційної безпеки

1.1.1 Політика безпеки

Політика безпеки[1] - це документ, що описує, як захистити організацію від загроз, включаючи загрози комп'ютерної безпеки, і як обробляти ситуації, коли вони відбуваються. Політика безпеки повинна ідентифікувати всі активи компанії, а також всі потенційні загрози для цих активів.

Політика безпеки повинна окреслити ключові елементи в організації, які необхідно захистити. Це може включати мережу компанії, її фізичну будівлю тощо. Вона також повинна окреслити потенційні загрози для цих елементів. Якщо документ зосереджується на кібербезпеці, загрози можуть включати в себе також можливість загрози зсередини, наприклад, незадоволені співробітники викрадуть важливу інформацію або запустять внутрішній вірус у мережі компанії. Ззовні організації хакери можуть проникнути в систему і спричинити втрату даних, зміни або їх викрадення. Нарешті, можуть виникнути фізичні пошкодження комп'ютерних систем.

Політика безпеки інформаційних технологій (IT) визначає правила та процедури для всіх осіб, які мають доступ до IT-ресурсів та ресурсів організації. Таким чином, ефективна політика безпеки IT є унікальним документом для кожної організації, що створюється відповідно до ризику втрати чи зливу інформації, що зберігається і передається, а також відповідно до цінності інформації.

1.1.2 Властивості безпеки інформації

Метою політики безпеки ІТ є збереження основних властивостей безпеки інформації, якими є:

- Конфіденційність передбачає захист активів від несанкціонованих доступу;
- Цілісність гарантує, що зміна активів здійснюється у визначеному та уповноваженому порядку;
- Доступність - це стан системи, в якій уповноважені користувачі мають постійний доступ до цих активів;
- Спостережність - підтримка спостережності ресурсів, систем, що дозволяє встановлювати суб'єкти, що проводять дії над об’єктами, а також оперативно реагувати на всі дії з метою мінімізації втрат.

1.1.3 Контроль доступу

Контроль доступу[2] - це фундаментальна концепція безпеки, яка регулює, хто або що може переглядати або використовувати ресурси в обчислювальному середовищі, для мінімізації ризику для бізнесу або організації.

Існує два типи контролю доступу: фізичний і логічний. Фізичний контроль доступу обмежує доступ до кампусів, будівель, приміщень і фізичних ІТ-активів. Логічне управління визначає підключення до системних даних і файлів, комп'ютерних мереж та інших логічних комп'ютерних ресурсів.

Щоб захистити дані, організації використовують електронні процедури контролю доступу, які використовують пристрої для читання карт доступу,

облікові дані користувачів, аудит і звіти, щоб відстежувати доступ працівників до певних кімнат і будівель, таких як центри обробки даних. Більшість систем контролю доступу використовують панелі керування доступом для обмеження доступу до будівель і приміщень, а також можливостей блокування та сигналізації для обмеження несанкціонованих операцій або доступу.

Системи контролю доступу використовують автентифікацію та авторизацію користувачів і організацій, оцінюючи необхідні реєстраційні дані, які можуть включати паролі, персональні ідентифікаційні номери (PIN-коди), біометричні сканування, маркери безпеки або інші фактори автентифікації. Багатофакторна автентифікація, яка вимагає двох або більше факторів автентифікації, часто є важливою частиною багаторівневого захисту для захисту систем контролю доступу.

Системи контролю доступу функціонують шляхом ідентифікації співробітника або юридичної особи, перевірки того, що ідентифікується співробітник або додаток, а також санкціонування набору дій та рівнів доступу, пов'язаних з обліковими даними. Протоколи та служби каталогів, такі як (LDAP) локальний протокол доступу до каталогів, пропонують механізми контролю доступу шляхом авторизації та автентифікації осіб і співробітників. Крім того, системи контролю доступу дозволяють користувачам використовувати організаційні ресурси, такі як веб-сервери та розподілені програми.

1.2 Операції реляційної алгебри

Реляційна алгебра є теоретичною мовою з операціями, які працюють на одному або декількох відношеннях, щоб визначити інше відношення без зміни початкового відношення.

Таким чином, як операнди, так і результати є відношеннями, і тому вихід з однієї операції може стати входом до іншої операції. Ця здатність дозволяє вкладати вирази в реляційну алгебру, так само, як ми можемо вкладати арифметичні операції.

Існує багато варіантів операцій, що входять до реляційної алгебри. П'ять основних операцій реляційної алгебри - вибірка, проекція, декартовий добуток, об’єднання і різниця множин - виконують більшість операцій з даними, які нас цікавлять. Крім того, існують також операції приєднання, перетину та розбиття, які можна виразити в термінах п'яти основних операцій. Функція кожної операції проілюстрована на рис. 1.1[3].

1.2.1 Унарні операції

Операція «Вибірка» (рис. 1.1, a) працює на одному відношенні R і визначає відношення, яке містить тільки ті кортежі R, які задовольняють заданій умові (предикату).

Операція «Проекція» (рис. 1.1, b) працює на одному відношенні R і визначає відношення, яке містить вертикальна підмножина R, витягуючи значення зазначених атрибутів і усуваючи дублікати.

1.2.2 Бінарні операції

Операції вибірки та проекції вибирають інформацію тільки з одного відношення. У випадках, коли ми хочемо використати інформацію з кількох відношень, використовуємо бінарні операції реляційної алгебри. Розглянемо операції декартового добутку(рис. 1.1, c), об'єднання(рис. 1.1, d), різниці(рис. 1.1, f) та перетину(рис. 1.1, e).

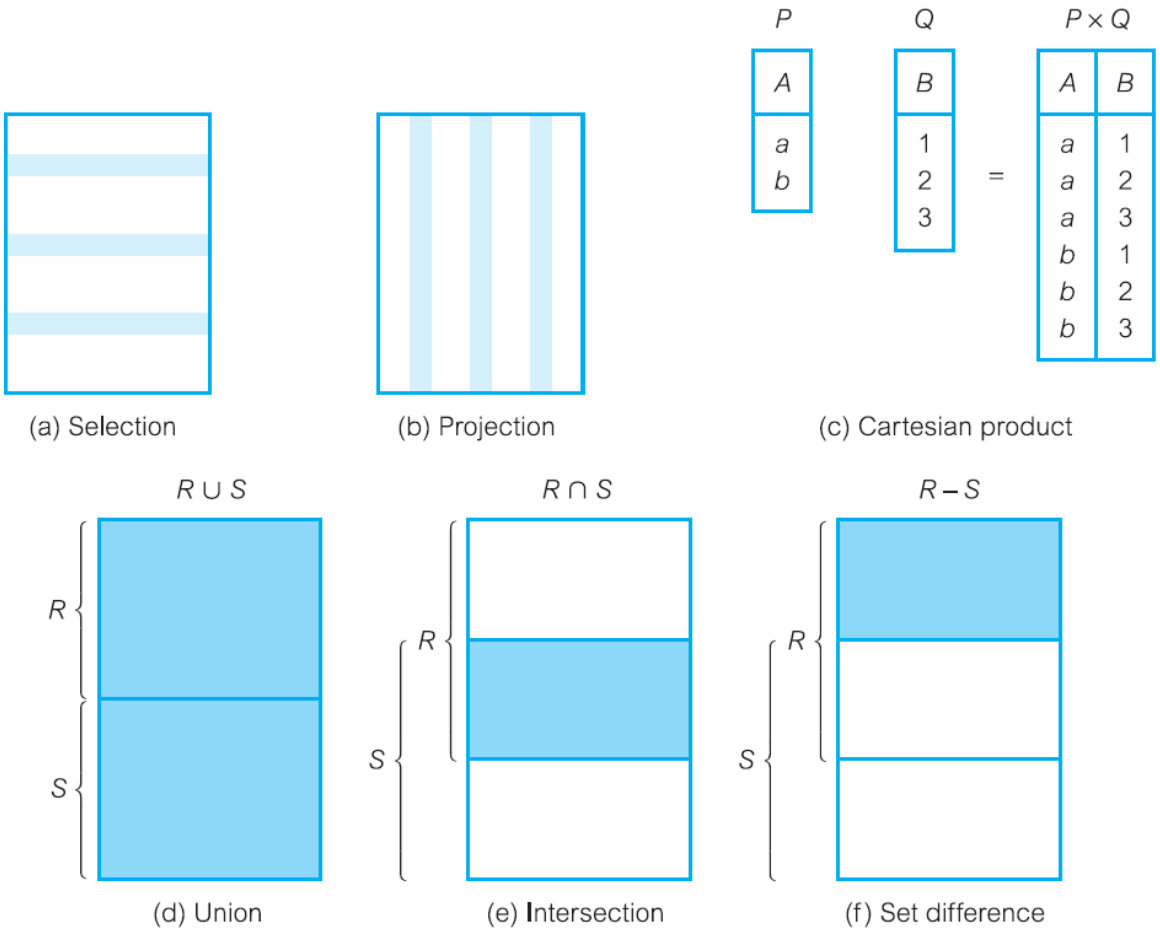

Рисунок 1.1 – Функції операцій реляційної алгебри

Операція декартового добутку R x S визначає відношення, що містить впорядковані пари <a, b>, де a належить R, а b належить S. Таке відношення складається з усіх можливих пар кортежів з двох відношень. Отже, якщо одне відношення має I кортежів і N атрибутів, а інше має J кортежів і M атрибутів, то декартовий добуток буде містити (I x J) кортежів з (N + M) атрибутами.

Об'єднання двох відношень R і S визначає відношення, яке містить всі кортежі R або S, або обидва R і S, дублюються кортежі, які усуваються. R і S повинні бути сумісними по об'єднанню. Якщо R і S мають I і J кортежі, то їх об'єднання виходить шляхом об'єднання їх в одне відношення з максимумом (I + J) кортежів.

Відношення називаються сумісними по об'єднанню, якщо:

- вони мають однакову кількість атрибутів з однаковими назвами, тобто для будь-якого атрибута в одному відношенні знайдеться атрибут з таким же найменуванням в іншому відношенні;
- атрибути з однаковими іменами визначені на одних і тих же типах даних.

Операція різниці множин визначає відношення, що складається з кортежів, що містить R, але не містить S. R і S повинні бути сумісними по об'єднанню.

Операція перетину визначає відношення, що складається з набору всіх кортежів, які знаходяться в обох R і S. R і S повинні бути сумісними по об'єднанню.

1.3 Рівні безпеки реляційних баз даних

Реляційна база даних (RDB) - це сукупність елементів даних з чітко визначеними відношеннями між ними[4]. Ці елементи організовані як набір таблиць з колонками і рядками.

Реляційна база даних складається з схеми відношення і екземпляра відношення. Схему відношення можна визначити як структуру реляційної бази даних, яка складається з імені відношення, імені кожного атрибута(або стовпця) і домену кожного атрибута. Схема відношення позначається як R (A_1; ...; A_n), де кожен A_i є атрибутом у реляційній схемі. Приклад відношення може бути визначений як набір записів, в якому кожен запис має таку ж кількість атрибутів, що і схема відношення.

Таблиці в базі даних логічно пов'язані, що полегшує пошук даних, організацію та звітність. RDB використовують SQL, яка є універсальною мовою для різних систем управління баз даних, що забезпечує простий інтерфейс програмування для взаємодії з базами даних.

Реляційні бази даних організовують дані різними способами. Кожна таблиця відома як відношення, яке містить одну або більше стовпців категорій даних. Кожен запис таблиці (або рядок) містить унікальний екземпляр даних, визначений для відповідної категорії стовпців. Одна або більше характеристик даних або запису відносяться до одного або декількох записів для формування функціональних залежностей. Вони класифікуються наступним чином:

- Один до одного: один запис таблиці стосується іншого запису в іншій таблиці.
- Один до багатьох: один запис таблиці пов'язаний з багатьма записами в іншій таблиці.
- Багато до одного: Більш ніж один запис таблиці стосується іншого запису таблиці.
- Багато для багатьох: Більш ніж один запис таблиці стосується більш ніж одного запису в іншій таблиці.

RDB виконує операції баз даних "select", "update" і "join", де select використовується для пошуку даних, update змінює дані, а join об'єднує їх комбінації.

RDB мають багато інших переваг, наприклад:

- Легко розширюється, оскільки нові дані можуть бути додані без зміни існуючих записів(також відома як масштабованість);
- Потужність і гнучкість;
- Безпека даних, які є критичними, коли обмін даними є конфіденційним. Наприклад, керівництво може поділяти певні привілеї даних і заблокувати доступ працівникам до даних, таких як конфіденційна інформація про зарплату або допомогу.

У реляційних базах даних для забезпечення безпеки бази даних можна використовувати декілька рівнів безпеки[5]. Рівні безпеки можуть бути класифіковані наступним чином:

• Автентифікація може бути визначена як концепція перевірки ідентичності користувача, який повинен мати доступ до реляційної бази даних. Кожен користувач повинен ідентифікувати себе перед тим, як мати доступ до даних, що зберігаються в системі реляційної бази даних. Автентифікація може відбуватися на різних рівнях, наприклад, автентифікація може виконуватися самою реляційною базою даних або іншим зовнішнім методам автентифікації користувачів.

• Керування доступом (авторизація) можна визначити як правила, які визначають, чи має користувач доступ до даних у відповідній базі даних. Правила авторизації керують зміною даних у реляційній базі даних. Управління доступом - це процедури, які визначаються для керування авторизацією даних у реляційній базі даних.

• Цілісність може бути визначена як група правил, які представляють правильний стан реляційної бази даних під час її модифікації.

• Аудит може бути визначений як відстеження всіх дій, що стосуються безпеки, визначених користувачем.

Механізм контролю доступу забезпечує конфіденційність даних. Кожен раз, коли суб'єкт намагається отримати доступ до об'єкта даних, механізм контролю доступу перевіряє права користувача щодо набору авторизацій, як правило, задається адміністратором безпеки. Контроль доступу гарантує, що всі прямі звернення до об'єктів бази даних відбуваються тільки відповідно до правил, що регулюються політикою безпеки.

1.4 Управління доступом в реляційних базах даних

Запобігання несанкціонованому доступу до реляційної бази даних є головною метою впровадження захищеної системи управління базами даних. Більшість користувачів баз даних потребують лише спеціального дозволу на деякі частини реляційної бази даних для виконання своїх завдань. Дозволити їм доступ до всієї бази даних небажано. Таким чином, політика безпеки повинна бути розроблена ефективно, щоб дозволити групі користувачів отримати доступ тільки до необхідних частин бази даних.

Як тільки політика безпеки буде розроблена, вона повинна бути виконана для досягнення необхідного рівня безпеки. Три основні підходи в СУБД для контролю доступу - дискреційний контроль доступу, мандатний контроль доступу та контроль доступу на основі ролей.

Підходи для здійснення контролю доступу в реляційних базах даних описані наступним чином:

• Керування доступом на основі представлень: відношення - це фізична локація в реляційній базі даних, яка зберігає дані в реляційній базі даних.

Представлення - це логічний набір збереженого запиту до бази даних. На відміну від фізичної таблиці реляційної бази даних, представлення є логічною таблицею, що динамічно створюється з даних у реляційній базі даних, коли до неї звертається запит.

• Модифікація запитів: запит, написаний користувачем, змінюється, щоб включати обмеження, визначені привілеями користувача.

Наприклад, DBA надає користувачеві A можливість вибирати тільки працівників, які знаходяться в краєзнавчому відділі, з таблиці співробітників. Коли A робить запит до таблиці, він отримує інформацію тільки про ті дані, які стосуються працівників даного відділу.

1.4.1 Дискреційний контроль доступу

Дискреційний контроль доступу(DAC) базується на наданні та відкликанні привілеїв для використання системних об'єктів (представлень, стовпців тощо). Привілеї надаються (або скасовуються) кожному суб'єкту (користувачеві, обліковому запису, програмі) окремо[6].

Політики дискреційного контролю доступу дозволяють розповсюджувати права доступу від одного суб'єкта до іншого. Це називається дискреційним у тому сенсі, що власник даних має повну свободу дій щодо надання / відкликання прав доступу до своїх даних. У DAC надання або відкликання привілеїв може виконуватися адміністратором бази даних (DBA).

Типи привілеїв DAC описані таким чином:

- Привілеї облікового запису: Кожен користувач має привілеї, які не залежать від відношень у базі даних. Наприклад, DBA надає / відкликає привілеї користувачеві для CREATE TABLE, CREATE VIEW, DROP і ALTER.

- Привілеї відношення: адміністратор баз даних може вказати привілей для зміни кожного окремого співвідношення у реляційній базі даних.

Наприклад, DBA надає/відкликає привілеї SELECT/MODIFY/ REFERENCE на конкретне відношення R.

- Дискреційні контролі доступу можуть надаватися багатьом об'єктам у системі реляційної бази даних, таким як база даних, група відношень, одне відношення, набір атрибутів одного відношення і група записів одного відношення.

DAC має деякі недоліки при застосуванні до реляційної бази даних:

- Застосування політики безпеки: використання DAC залежить від концепції володіння даними в підприємстві. В DAC користувач, який створює об'єкт у реляційній базі даних, є власником цього об'єкта і може надавати доступ іншим користувачам до цього об'єкта. Недолік цього полягає в тому, що підприємство не може забезпечувати виконання своїх вимог щодо безпеки, бо не може контролювати дії всіх користувачів, які створюють об'єкти в реляційній базі даних та надають дозволи на них.
- Каскадна авторизація: розглянемо ситуацію, коли є три користувачі: U1, U2 і U3. Користувач U2 має привілей на об'єкт O від U1 і надає цей привілей U3. Пізніше U1 надає привілей U3 на об'єкт O, але U2 з певних причин відкликає привілей від привілеїв U3. В результаті цих операцій U3 все ще має права доступу (від U1) до об'єкта O, хоча U2 скасував йому можливість отримання доступу.

- Атаки троянських коней: Троянський кінь може використовуватися для надання певного привілею користувача об'єкту іншому користувачеві, не маючи жодної інформації про цього користувача.
- Проблеми оновлення: У DAC захист на основі представлення є логічним запитом, який не має фізичних даних у реляційній базі даних. Недоліком захисту на основі представлення є те, що не всі дані можуть бути оновлені.

1.4.2 Мандатний контроль доступу

Мандатний контроль доступу (MAC) - це метод обмеження доступу користувачів до об'єктів, які містять певну конфіденційну інформацію[7].

MAC залежить від рівня безпеки, пов'язаного з кожним об'єктом у відповідній базі даних і кожному користувачеві. Рівень безпеки на об'єкті визначається як класифікація безпеки, тоді як рівень безпеки для користувача визначається як дозвіл на безпеку. Реалізація MAC - багаторівнева безпека(MLS), яка була розроблена в основному для комп'ютерних систем та систем баз даних в урядових організаціях, таких як органи розвідки або Міністерство оборони.

Мандатний контроль доступу оцінює відношення домінування між мітками безпеки користувача та мітками безпеки об'єкта і визначає, чи дозволяти певні дії на основі певних правил:

- Якщо мітка захисту користувача домінує над міткою безпеки об'єкта, користувач може читати об'єкт.
- Якщо мітка захисту користувача і мітка захисту об'єкта є еквівалентними, користувач може читати об'єкт і записувати в нього.

- Якщо мітка захисту користувача домінує над міткою безпеки об'єкта, користувач не може записати об'єкт.
- Якщо мітка захисту користувача не може бути порівняна з міткою безпеки об'єкта, користувач не може читати або записувати в цей об'єкт.

Ієрархічні рівні безпеки використовуються як основа для прийняття рішень в мандатному контролі доступу. При визначенні рівня безпеки об'єкта визначається ступінь «чутливості» цього об'єкта. Рівні безпеки забезпечують захист об'єкта певного рівня безпеки від доступу користувача нижчого рівня.

1.4.3 Контроль доступу на основі ролей

Головною метою для рольового контролю доступу (RBAC) є необхідність моделювати структуру природної політики для безпеки організації. RBAC базується на ролях, які мають користувачі. Ролі подібні до ролей груп користувачів у контролі доступу.

У RBAC роль визначається як група дій і обов'язків, що належать до певної діяльності. Роль може представляти роботу користувача (наприклад, покупця), або вона може визначити дію, яку повинен виконати користувач (наприклад, замовити матеріал). Замість визначення всіх дозволів для кожного з користувачів, які виконують одне і те ж завдання, для ролей можуть бути визначені дозволи на об'єкти. Користувачеві, якому присвоєно роль, можна виконати всі дії, на які вона має дозвіл. Компоненти RBAC можна описати наступним чином[8]:

- Відношення роль-дозвіл: цей компонент керує наданням/відкликанням дозволу на певну роль.

- Відношення користувач–роль: цей компонент визначає, як призначити користувачів певній ролі.
- Відношення роль-роль: цей компонент визначає, як зробити роль членом іншої ролі.

RBAC має три принципи безпеки:

- Найменший привілей: RBAC дозволяє користувачеві отримувати доступ до об'єктів з найменшим привілеєм, необхідним для певного завдання, яке необхідно виконати. Це мінімізує атаку троянських коней.
- Розподіл обов'язків: RBAC гарантує, що жоден користувач не має достатньо привілеїв для самостійного використання системи.
- Абстракція даних: Це підтримується за допомогою абстрактних привілеїв, таких як кредит і дебет рахунку.

Висновки до розділу 1

В цьому розділі було проведено огляд безпеки реляційних баз даних та управління доступом в них, а також недоліки типів управління доступом при застосуванні до реляційної бази даних. Було з’ясовано, що дискреційне управління доступом не підходить для реалізації систем, адже має вагомі недоліки у застосуванні до баз даних в сучасних системах. Завдяки аналізу існуючих підходів та механізмів захисту від несанкціонованого доступу до реляційної бази даних, були виявлені необхідні складові для подальшої розробки систем багаторівневої безпеки БД. Такі системи є реалізацією мандатного управління доступу, що дозволяє надійно та гнучко розмежовувати доступ до об’єктів різного ступеню важливості.

2 БАГАТОРІВНЕВА БЕЗПЕКА БАЗ ДАНИХ

2.1 Принципи та переваги багаторівневої безпеки

Багаторівнева безпека - це політика безпеки, яка дозволяє класифікувати об'єкти та користувачів на основі системи ієрархічних рівнів безпеки та системи неієрархічних категорій безпеки[9].

Багаторівнева безпека забезпечує можливість запобігання доступу неавторизованих користувачів до інформації за більш високою класифікацією, ніж їх авторизація.

Багаторівнева безпека була створена в США, для використання військовими. Розглянемо наступну ситуацію: припустимо, є база даних, яка містить певну конфіденційну інформацію. Конфіденційні дані класифікуються на різні рівні безпеки і повинні бути оброблені на спеціалізованих системах, які не надають доступу користувачам за межами передбаченого рівня безпеки. Основні обмеження для такої системи можна описати наступним чином:

- Резервні бази даних: Для зберігання даних у реляційній базі даних на різних рівнях безпеки слід створювати іншу базу даних для кожного рівня безпеки.
- Резервні робочі станції: для отримання кожного типу даних необхідно мати різні робочі станції.
- Висока вартість IT-інфраструктури: існує ризик спільного використання мережевих ресурсів.

- Неефективність: Користувачі повинні отримати привілеї на декількох системах реляційних баз даних для виконання своїх обов'язків.

Багаторівнева безпека – чудове рішення для забезпечення належного захисту для конфіденційної інформації. MLS надає доступ до даних у різних рівнях класифікації безпеки користувачам, які мають різні рівні захисту. У багаторівневій безпеці кожен елемент даних визначається як об'єкт і має рівень класу безпеки, кожен користувач визначається як суб'єкт і також має власний рівень класу безпеки. Рівень класу безпеки об'єкта або суб'єкта A називається міткою і позначається як L(A).

Контроль доступу в багаторівневій безпеці базується на моделі Белла-ЛаПадули[10], яка має такі властивості(формули 2.1, 2.2, 2.3 відповідно):

- Проста властивість безпеки: користувачеві s дозволяється доступ для читання об'єкта o, тільки якщо рівень суб’єкта L(s) вище або дорівнює рівню об’єкта L(o).

$$\forall\, s \in S,\ \forall\, o \in O,\ read\ \in A[s,o] \rightarrow\ L(s) \geq L(o) \quad (2.1)$$

- Властивість * : користувачеві s дозволяється доступ до запису об'єкта o, тільки якщо L(s) нижче або дорівнює L(o).

$$\forall\, s \in S,\ \forall\, o \in O,\ write\ \in A[s,o] \rightarrow\ L(s) \leq L(o) \quad (2.2)$$

- властивість strong *: користувачеві s дозволяється доступ до запису об'єкта o, тільки якщо рівень суб’єкта L(s) дорівнює рівню об’єкта L(o).

$$\forall\, s \in S,\ \forall\, o \in O,\ write\ \in A[s,o] \rightarrow\ L(s) = L(o) \quad (2.3)$$

В формулах 2.1 – 2.3: S – множина суб’єктів, O – множина об’єктів, A[s,o] – матриця доступу, L(s) – рівень безпеки суб’єкту, L(o) – рівень безпеки об’єкту.

Багаторівнева безпека пропонує наступні переваги:

- Багаторівневе забезпечення безпеки є обов'язковим і автоматичним.
- Багаторівнева безпека може використовувати методи, які важко виражати через традиційні SQL-представлення або запити.
- Багаторівнева безпека не покладається на спеціальні представлення або змінні бази даних для забезпечення контролю безпеки на рівні рядків.
- Багаторівневі засоби безпеки узгоджені та інтегровані по всій системі, так що ви можете уникнути визначення користувачів і дозволів більше одного разу.
- Багаторівнева безпека не дозволяє користувачам розсекречувати інформацію.

В багаторівневій моделі реляційної БД кожен елемент даних позначається власним класифікатором безпеки. Ієрархія багаторівневої безпеки має чотири рівні підвищення «чутливості». Ці рівні, від найнижчого до найвищого, - некласифікований(U), конфіденційний(C), секретний(S) і цілком секретний(TS). Для отримання доступу до даних користувачі повинні мати відповідний класифікатор безпеки.

Багаторівнева безпека обмежує доступ до об'єкта або запису на основі мітки захисту об'єкта або запису і мітки захисту користувача.

Для локальних підключень мітка захисту користувача - це мітка захисту, вказана користувачем під час входу. Якщо під час входу не вказано мітку захисту, мітка захисту за замовчуванням є міткою захисту користувача.

Для звичайних з'єднань TCP/IP мітку безпеки користувача може визначати зона безпеки. IP-адреси зазвичай згруповані в зони безпеки на сервері. Для надійних підключень TCP/IP мітка захисту користувача є міткою захисту, встановленою в довіреному контексті.

Для з'єднань SNA замість мітки захисту, з якою користувач увійшов, використовується мітка захисту за замовчуванням для користувача.

Мітки безпеки можуть бути призначені користувачеві шляхом встановлення надійного з'єднання в довіреному контексті. Визначення надійного контексту визначає мітку захисту, пов'язану з користувачем у надійному з'єднанні.

В багаторівневій моделі реляційної бази даних відношення може бути багатозначне – коли воно містить два або більше записів з однаковими значеннями первинного ключа. Багатозначність(polyinstantiation) виникає в наступних двох ситуаціях[11]:

- Невидима багатозначність може відбуватися, коли користувач з низьким рівнем безпеки вставляє дані в атрибут, який вже містить дані з більш високим рівнем безпеки.

- Видима багатозначність може відбутися, коли користувач з високим рівнем безпеки вставляє дані в атрибут, який вже містить дані на нижчому рівні безпеки.

Існує два різних типи багатозначності[12]:

- Багатозначність сутності:

У багаторівневій реляційній базі даних багатозначність сутності може виникнути, коли відношення містить більше одного запису з однаковими

значеннями первинного ключа, але з різними значеннями класу доступу для первинного ключа. Наприклад, є два записи з одним і тим же первинним ключем, але з двома різними класами безпеки.

- Багатозначність атрибуту:

У багаторівневій реляційній базі даних багатозначність атрибуту може відбуватися, коли відношення містить два або більше записів з ідентичним первинним ключем і його значеннями рівня безпеки, але з різними значеннями для одного або декількох залишилися атрибутів.

2.2 Огляд моделей із багаторівневою безпекою БД

Існує декілька багаторівневих моделей безпеки реляційних баз даних, наприклад, SeaView і запропоновані Sandhu–Jajodia, Smith–Winslett і т.д. Розглянемо застосування різних моделей багаторівневої безпеки на таблиці, що містить дані про працівників(табл. 2.1)

Таблиця 2.1 – Початкова таблиця

Працівник (Primary Key)	Відділ	Зарплата
Андрій Василенко (U)	SMM (U)	8000 (U)
Андрій Василенко (S)	Sales (S)	20000 (S)

2.2.1 Модель SeaView

В моделі безпечних представлень даних (Secure data views) рівні безпеки призначаються кожному елементу даних, для кожного атрибута запису у відношенні, як показано в таблиці 2.1. У моделі SeaView дані зберігаються в наборі однорівневих фрагментів, і багаторівневі відношення реалізуються як представлення цих однорівневих відношень.

В реалізації моделі SeaView використовуються два алгоритми:

• Алгоритм декомпозиції розділяє багаторівневе відношення на однорівневі фрагменти.

• Алгоритм відновлення відновлює початкове багаторівневе відношення.

Застосовуючи дану модель до таблиці 2.1, отримуємо дані у вигляді 5 однорівневих фрагментів - один primary key та 4 групи відношень(табл. 2.2).

Таблиця 2.2 – Застосування моделі SeaView

Працівник (Primary Key)	Відділ
Андрій Василенко (U)	SMM (U)

Працівник (Primary Key)	Зарплата
Андрій Василенко (U)	8000 (U)

Працівник (Primary Key)	Відділ
Андрій Василенко (U)	Sales (S)

Працівник (Primary Key)	Зарплата
Андрій Василенко (U)	20000 (S)

2.2.2 Модель Jajodia–Sandhu

Модель Jajodia–Sandhu є модифікацією моделі SeaView. Дана модель має модифікований алгоритм, який розкладає багаторівневе відношення на фрагменти з одним рівнем, а також модифікує алгоритм відновлення, що відновлює початкове багаторівневе співвідношеннях[13].

У моделі Jajodia–Sandhu алгоритм декомпозиції використовує тільки горизонтальну фрагментацію, оскільки вертикальні фрагментації не потрібні. Це призводить до поліпшення алгоритму відновлення, оскільки можна реконструювати багаторівневе відношення без виконання операцій приєднання, для реконструкції багаторівневого відношення потрібні тільки операції об’єднання(union).

Наприклад, відношення з Таблиці 2.1 буде розкладено на два однорівневих фрагмента(Таблиця 2.3).

Таблиця 2.3 – Застосування моделі Jajodia–Sandhu

Рівень безпеки	Працівник (Primary Key)	Відділ	Зарплата
U	Андрій Василенко (U)	SMM (U)	8000 (U)

Рівень безпеки	Працівник (Primary Key)	Відділ	Зарплата
S	Андрій Василенко (U)	Sales (S)	20000 (S)

2.2.3 Модель Smith–Winslett

В моделі Smith–Winslett багаторівнева реляційна база даних розглядається як набір звичайних реляційних баз даних, де всі бази даних мають однакову схему. Ця модель не підтримує безпеку на рівні кожного окремого атрибута. Рівень безпеки може бути призначений тільки для атрибутів первинних ключів і для запису в цілому. В даній моделі користувач може бачити записи з рівнем безпеки, що не перевищує його власний.

Багаторівнева реляційна схема задається як R (TC, A_{PK}, C_{PK}, A_1 ..., A_n), де A_{PK} - атрибут первинного ключа, C_{PK} – атрибут-класифікатор первинного ключа, який містить рівень безпеки первинного ключа, набором A_1 ... A_n позначаються атрибути даних, а TC – атрибут-класифікатор даного запису, що містить його рівень безпеки. Використовуючи дану модель, відношення з Таблиці 2.1 буде інтерпретоване на відношення в таблиці 2.4.

Таблиця 2.4 – Застосування моделі Smith–Winslett

Рівень безпеки	Працівник (Primary Key)	Відділ	Зарплата
U	Андрій Василенко (U)	SMM	8000
S	Андрій Василенко (S)	Sales	20000

Модифікація БД(вставка, видалення та оновлення) може використовуватись користувачем лише на рівні безпеки користувача. Запит від користувача на рівні безпеки L може отримати доступ до даних саме з тих баз даних, рівень об’єктів яких не перевищує рівень L.

Модель Smith–Winslett також відома як модель семантики на основі переконання. В ній також введена концепція базового кортежу. Базовий кортеж - це найнижчий рівень безпеки запису бази даних, де існує захищеність сутності. Таким чином, процедура оновлення усуває проблеми, наявні в моделі Jajodia–Sandhu, але обмежує область оновлення до однієї сутності.

2.2.4 Модель Multilevel Relational

Багаторівнева реляційна модель представляє концепцію цілісності даних, яка забезпечує висхідний потік інформації. Модифікації даних на більш низькому рівні безпеки можуть бути автоматично передані до більш високих рівнів безпеки, які потребують запозичення цих даних.

Ця модель пов'язана з усуненням семантичної проблеми неоднозначності в моделі Jajodia–Sandhu. Користувач з рівнем безпеки може приймати дані, які складаються з двох частин: дані, що мають однаковий рівень безпеки, і дані, які запозичені від користувачів з нижчим рівнем безпеки. Дані, які суб'єкт може побачити, - це ті, що приймаються суб'єктами на рівні даних або на рівнях, нижчих за них.

Багаторівнева реляційна схема наведена як R(TC, A_{PK}, C_{PK}, A_1... $A_{n,}$, C_1 ... C_n), де A_{PK} позначається атрибут первинного ключа відношення, C_{PK} є атрибутом-

класифікатором первинного ключа, який містить його рівень безпеки, набір A_1 ... A_n - атрибути відношення, C_1 ... C_n – класифікатори для атрибутів, які містять їх рівень безпеки, а TC – атрибут-класифікатор запису, що містить його рівень безпеки.

У моделі даних MLR є декілька властивостей цілісності, з яких цілісність сутності та цілісність зовнішнього ключа взяті з оригінальної моделі SeaView, а цілісність багатозначності та референційна цілісність є суттєво переписаними авторами. Також вперше введена цілісність запозичених даних(data-borrow) [14]. Зокрема, властивість цілісності багатозначності, що наведена в даній моделі, є більш загальною, ніж у моделі SeaView або Jajodia-Sandhu, оскільки вона відповідає і за багатозначність елементів. Переглянута референтна властивість цілісності також має справу з деякими новими опорними проблемами, що виникають внаслідок запозичення даних.

В таблиці 2.5 можна побачити, як користувач з рівнем безпеки S скористався командою UPLEVEL, щоб вказати, що він довіряє інформації з першого запису. Для цього він вставляє другий ідентичний запис, тільки з класифікатором безпеки запису на своєму рівні S.

Таблиця 2.5 – Застосування моделі MLR

Рівень безпеки	Працівник (Primary Key)	Відділ	Зарплата
U	Андрій Василенко (U)	SMM	8000
S	Андрій Василенко (U)	SMM	8000

2.3 Алгоритми операцій для впровадження моделей

Розглянемо алгоритми операцій з даними, що використовуються для впровадження моделей багаторівневої безпеки БД.

2.3.1 Операція SELECT

- Модель SeaView

Операція вибірки в моделі SeaView реалізується наступним чином(рис. 2.1):

Рисунок 2.1 – Операція Select в моделі SeaView

Спершу визначаємо рівень безпеки користувача, який виконує операцію. Далі з даних, що зберігаються у вигляді однорівневих відношень, створюється представлення шляхом виконання операції з'єднання між вертикальними відношеннями і об'єднанням між горизонтальними однорівневими відношеннями. В результаті користувач отримує дані з цього представлення, рівень безпеки первинного ключа яких нижче або дорівнює рівню користувача.

- Модель Jajodia–Sandhu

Алгоритм операції вибірки в даній моделі продемонстровано на блок-схемі на рис. 2.2. Після того, як користувач з певним рівнем безпеки виконує операцію, із горизонтальних однорівневих відношень шляхом об'єднання створюється представлення. Користувач отримає набір рядків з цього представлення, рівень первинного ключа яких не перевищує його власний.

- Моделі Smith–Winslett та MLR

Алгоритм впровадження операції вибірки в даних моделях, як і в інших, починається з визначення рівня безпеки користувача. На відміну від попередніх моделей, в цій не потрібне формування представлення, адже дані не розділяються на однорівневі відношення. Також є відмінність в тому, що операція вибірки в даній моделі базується на класифікаторі запису. Тобто користувач отримає набір рядків, рівень безпеки запису яких не перевищує його власний.

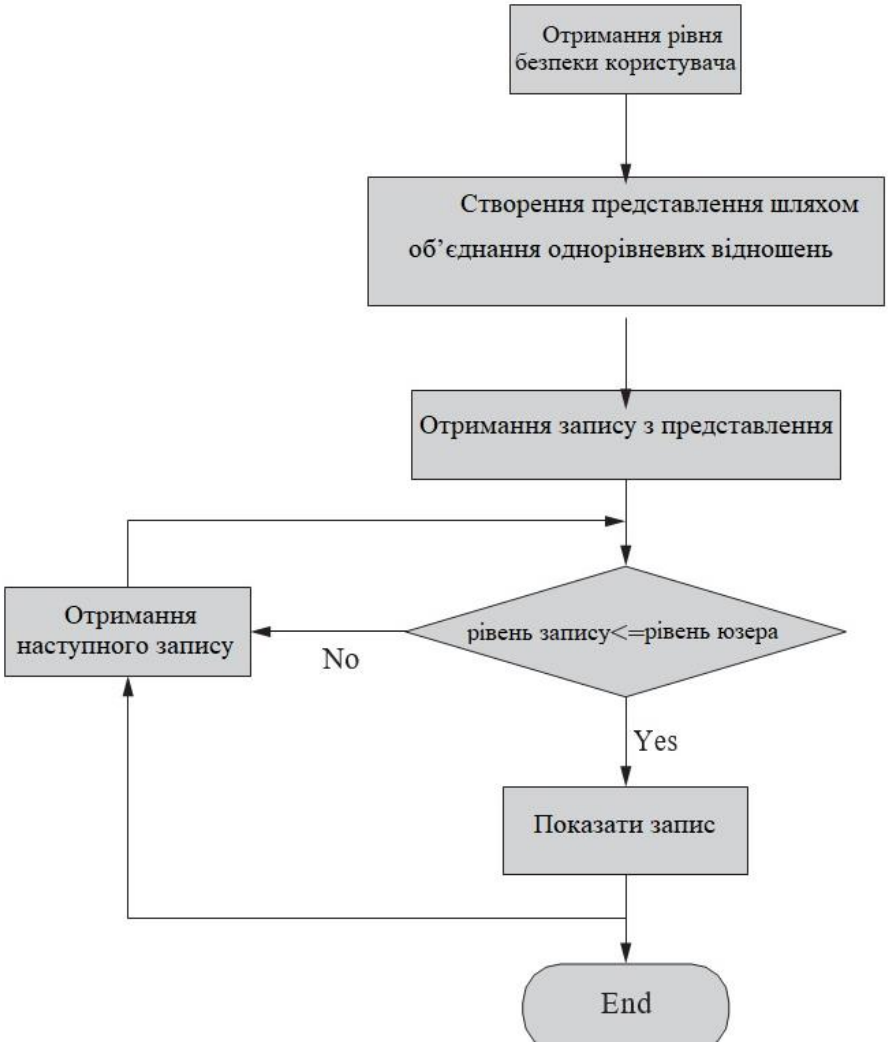

Рисунок 2.2 – Операція Select в моделі Jajodia-Sandhu

Алгоритм операції вибірки в моделях Smith–Winslett та MLR продемонстровано на блок-схемі на рис. 2.3.

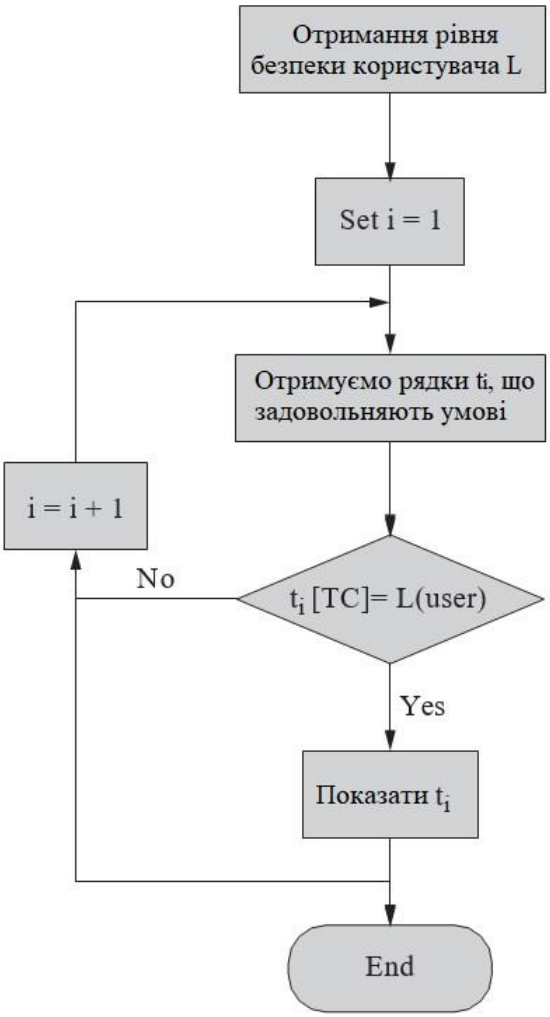

Рисунок 2.3 – Операція Select в моделі Smith-Winslett та MLR

2.3.2 Операція INSERT

- Моделі SeaView та Jajodia-Sandhu

Операція вставки виконується наступним чином(рис. 2.4): перевіряємо рівень безпеки користувача, який виконує операцію, та наявність атрибуту в списку атрибутів, що фігурують в операції. Рівень безпеки цих атрибутів буде

встановлений на рівні безпеки користувача. Далі в однорівневі відношення вставляємо значення, що відповідають атрибутам цих однорівневих відношень, і присвоюємо їм рівень, що дорівнює рівню користувача.

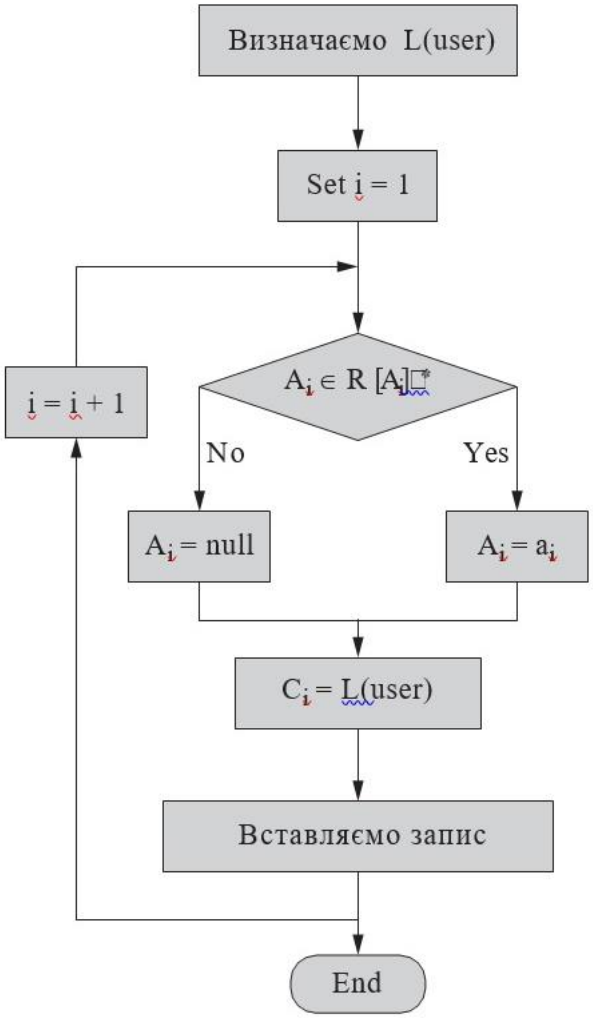

Рисунок 2.4 – Операція Insert в моделях SeaView та Jajodia-Sandhu

- Модель Smith–Winslett

В даній моделі запис буде вставлений в багаторівневе відношення, а рівень безпеки первинного ключа встановлюється на рівні безпеки користувача, який

виконує операцію. Алгоритм виконання операції вставки в даній моделі продемонстровано на блок-схемі на рис. 2.5.

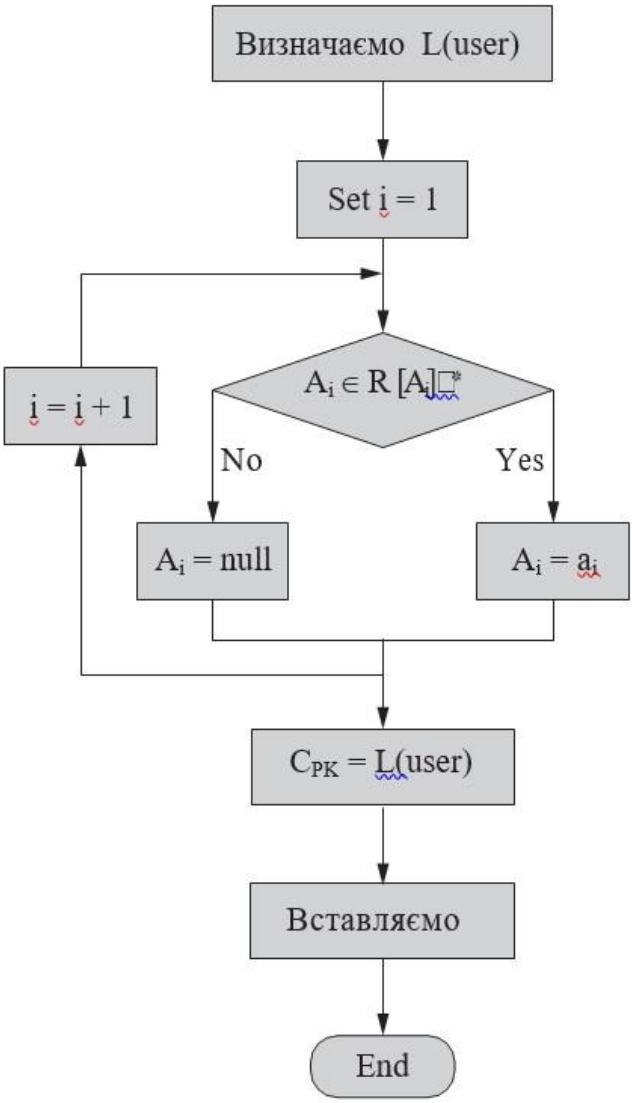

Рисунок 2.5 – Операція Insert в моделі Smith-Winslett

- Модель MLR

В даній моделі запис буде вставлений в багаторівневе відношення, а рівень безпеки всіх атрибутів встановлюється на рівні безпеки користувача, який виконує операцію. Алгоритм виконання операції вставки в даній моделі продемонстровано на блок-схемі на рис. 2.6.

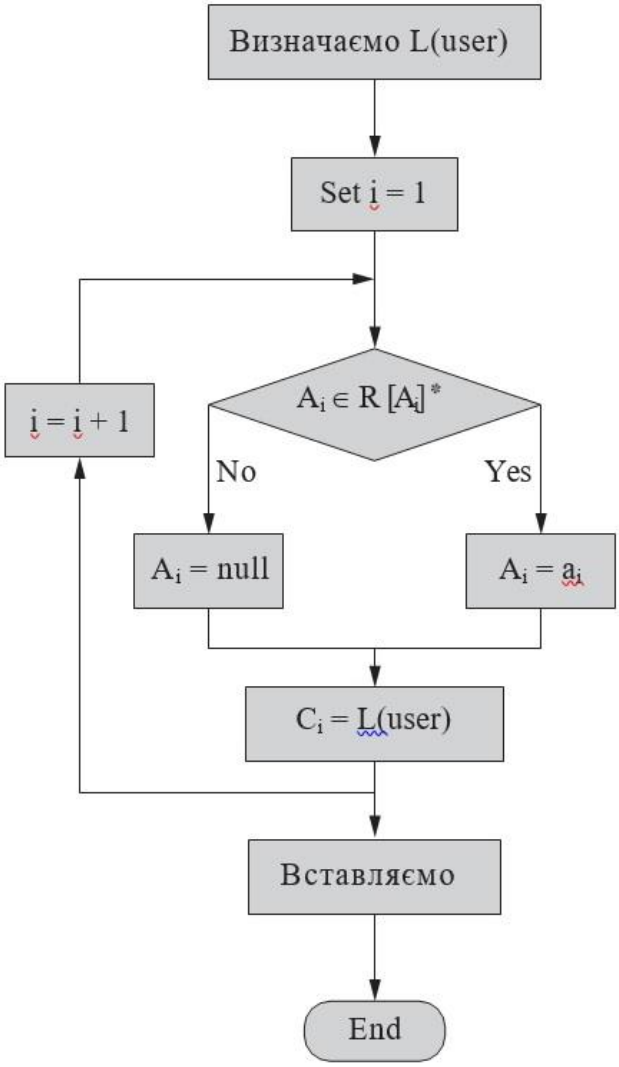

Рисунок 2.6 – Операція Insert в моделі MLR

2.3.3 Операція UPDATE

- Моделі SeaView та Jajodia–Sandhu

Алгоритм операції оновлення в даних моделях продемонстрований на рис. 2.7. Спершу перевіряємо рівень безпеки користувача, який виконує операцію, та обираємо рядки, що задовольняють умові оновлення та мають рівень безпеки, що не перевищує рівень безпеки користувача.

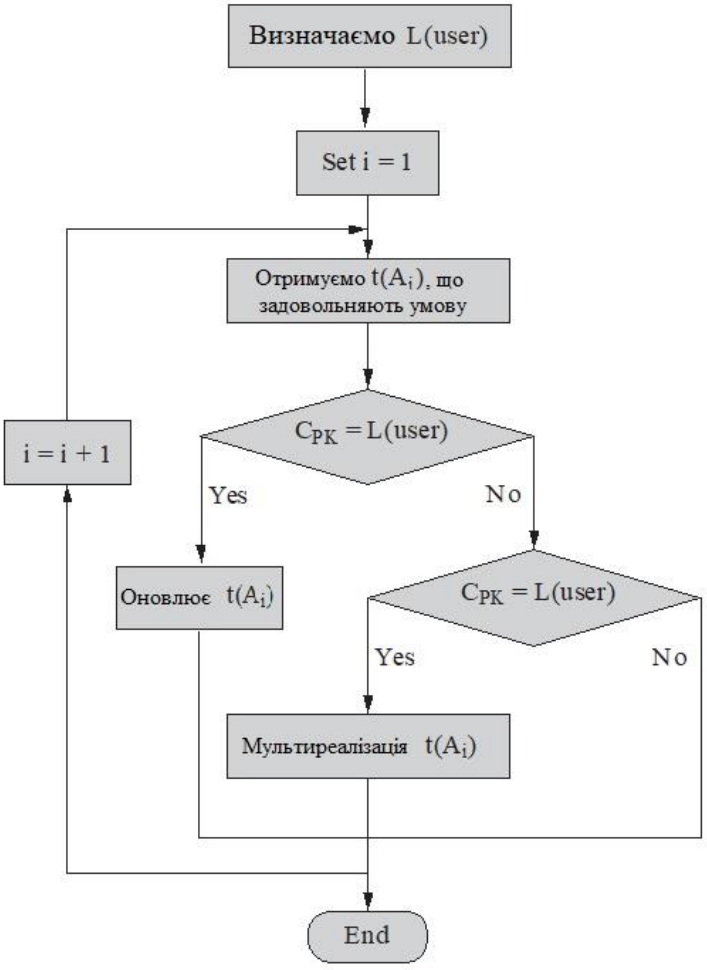

Рисунок 2.7 – Операція оновлення в моделях SeaView та Jajodia–Sandhu

Далі поділяємо отримані рядки на ті, рівень безпеки первинного ключа яких дорівнюють рівню користувача, і ті, рівень яких менше. Для рядків з таким же рівнем дані у відношенні будуть оновлені, а для рядків з меншим рівнем - рівень безпеки в однорівневих відношеннях з вказаними атрибутами підвищено до рівня безпеки користувача, і записи додано у відношення, тобто виникає багатозначність.

- Модель Smith–Winslett

Алгоритм операції оновлення в даній моделі(на рис. 2.8) відрізняється від алгоритму для попередніх моделей.

Як і в операції вибору, перевірка рівня безпеки базується на перевірці класифікатора запису. Коли користувач з певним рівнем безпеки виконує цю операцію, оновлюватимуться лише ті записи, що відповідають умові та класифікатор яких дорівнює рівню безпеки користувача.

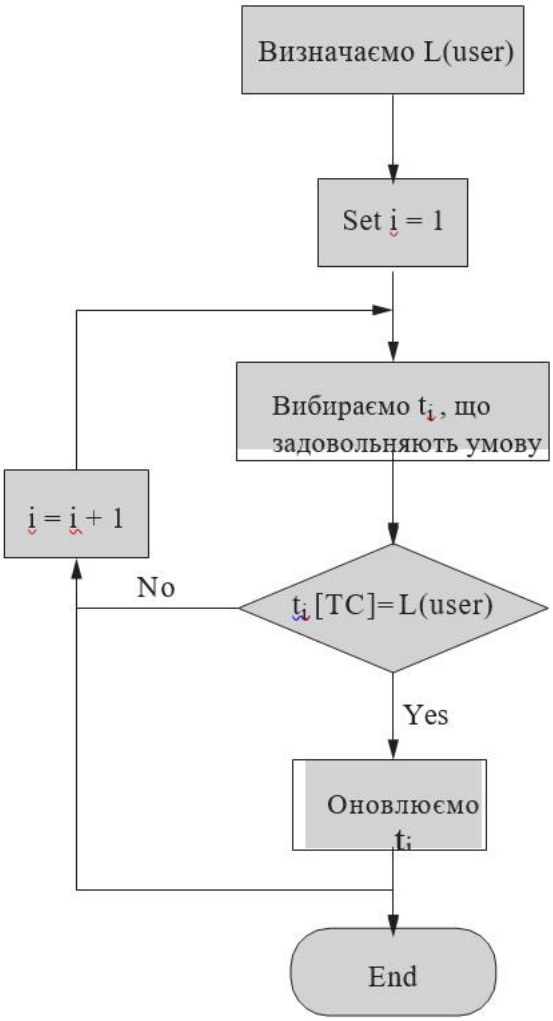

Рисунок 2.8 – Операція оновлення в моделі Smith-Winslett

- Модель MLR

Алгоритм виконання операції оновлення в даній моделі продемонстровано на блок-схемі на рис. 2.9.

Операція оновлення в даній моделі виконується наступним чином:

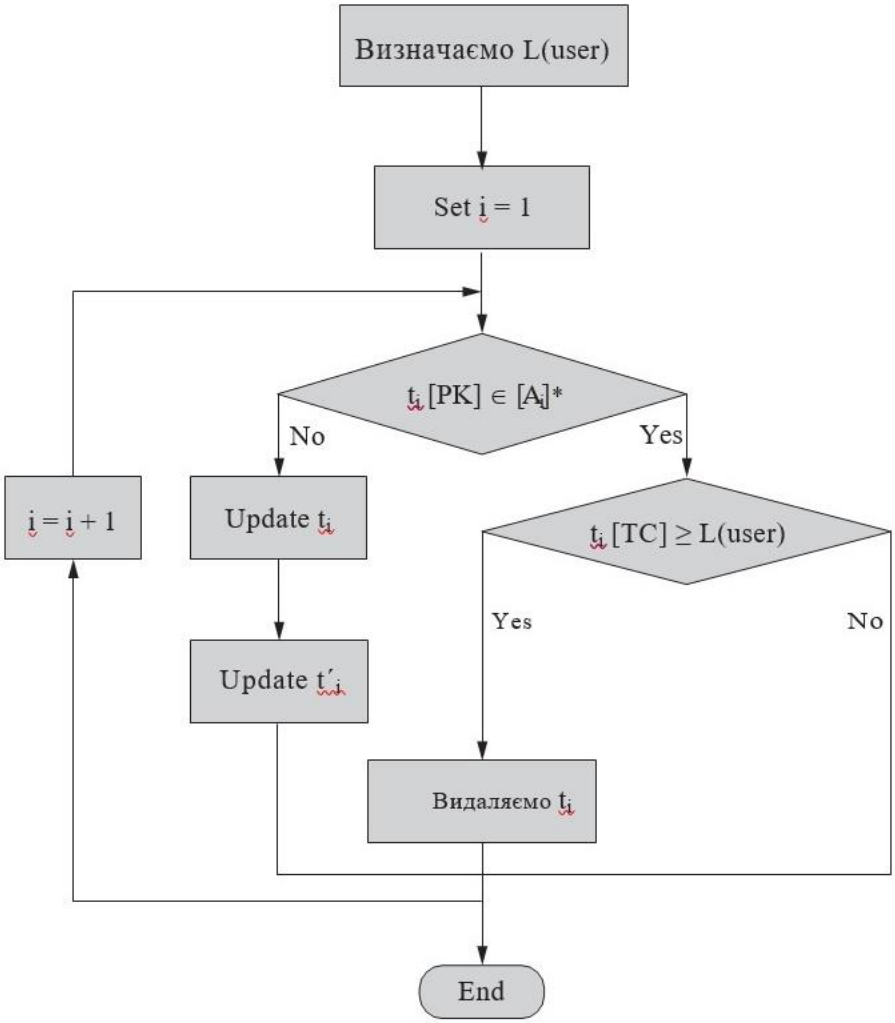

Рисунок 2.9 – Операція оновлення в моделі MLR

Спершу за умови, що атрибут первинного ключа не знаходиться в SET, оновлюються всі записи в багаторівневому відношенні, які задовольняють умові оновлення, і мають класифікатор запису, що дорівнює рівню безпеки користувача. Крім того, усі запозичені записи користувачів вищого рівня, які задовольняють умові оновлення, також будуть оновлені.

Далі перевіряємо, якщо деякий атрибут первинного ключа знаходиться в SET, то оновлюються всі записи в багаторівневому відношенні, які задовольняють умові P оновлення і мають класифікатор запису, рівний рівню безпеки користувача. При цьому видаляються всі запозичені записи користувачами вищого рівня, які задовольняють умові оновлення.

2.3.4 Операція DELETE

- Моделі SeaView та Jajodia–Sandhu

Коли користувач виконує операцію видалення, спершу визначається його рівень безпеки. Далі з однорівневих відношень видаляються всі записи, які задовольняють заданій в операції умові та мають рівень безпеки первинного ключа, що дорівнює рівню користувача.

Алгоритм виконання операції видалення в даних моделях продемонстровано на блок-схемі на рис. 2.10.

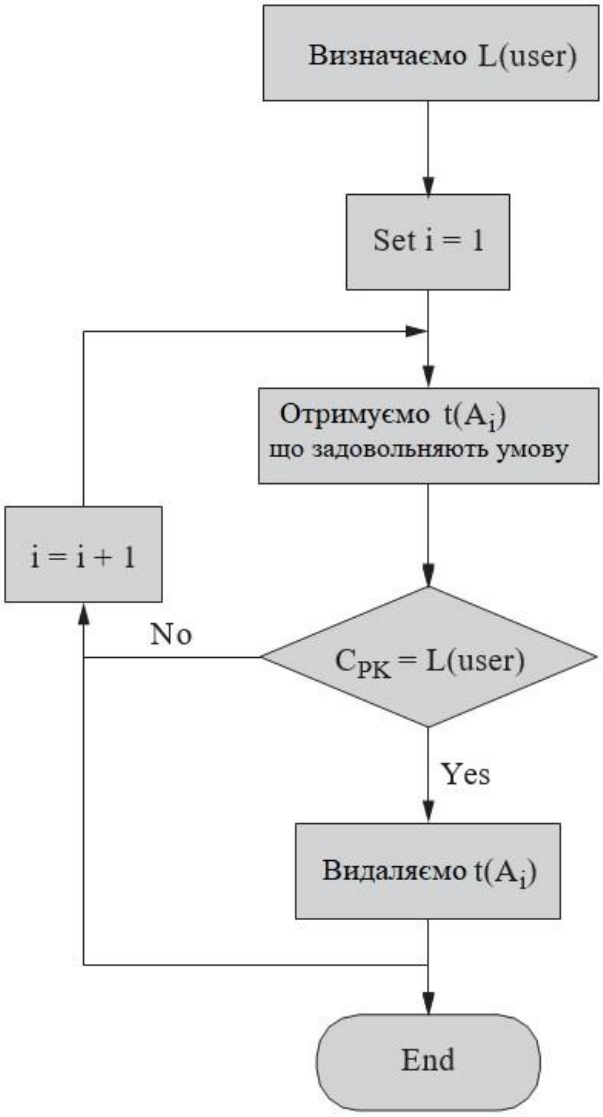

Рисунок 2.10 – Алгоритм видалення в моделях SeaView та Jajodia–Sandhu

- Модель Smith–Winslett

Як і решта операцій в даній моделі, операція видалення базується на перевірці класифікатора рядку. Алгоритм виконання операції видалення в даній моделі продемонстровано на блок-схемі на рис. 2.11. Коли користувач виконує

операцію видалення, визначається його рівень безпеки. Далі з багаторівневого відношення видаляються лише ті записи, які задовольняють заданій умові та мають класифікатор запису, що дорівнює рівню безпеки користувача.

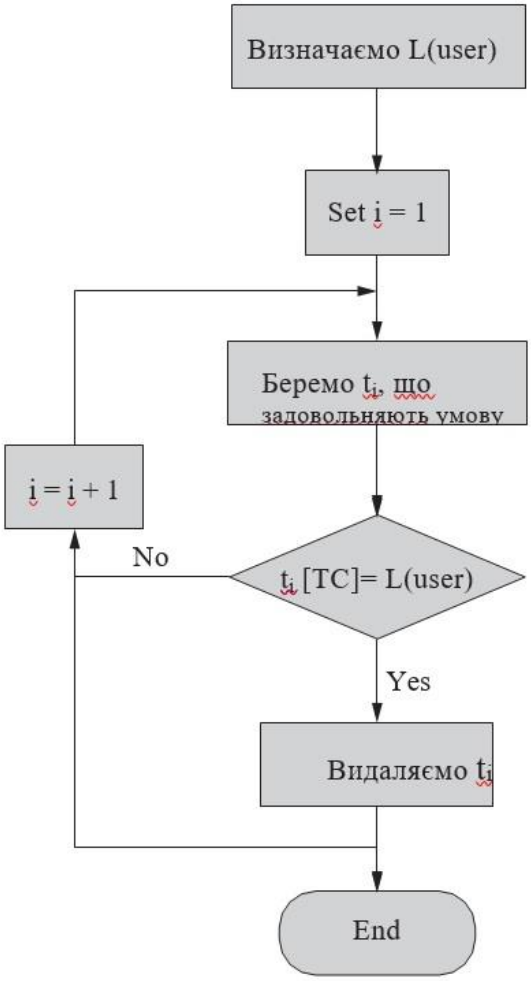

Рисунок 2.11 – Алгоритм видалення в моделі Smith–Winslett

- Модель MLR

Алгоритм виконання операції видалення в даній моделі продемонстровано на блок-схемі на рис. 2.12.

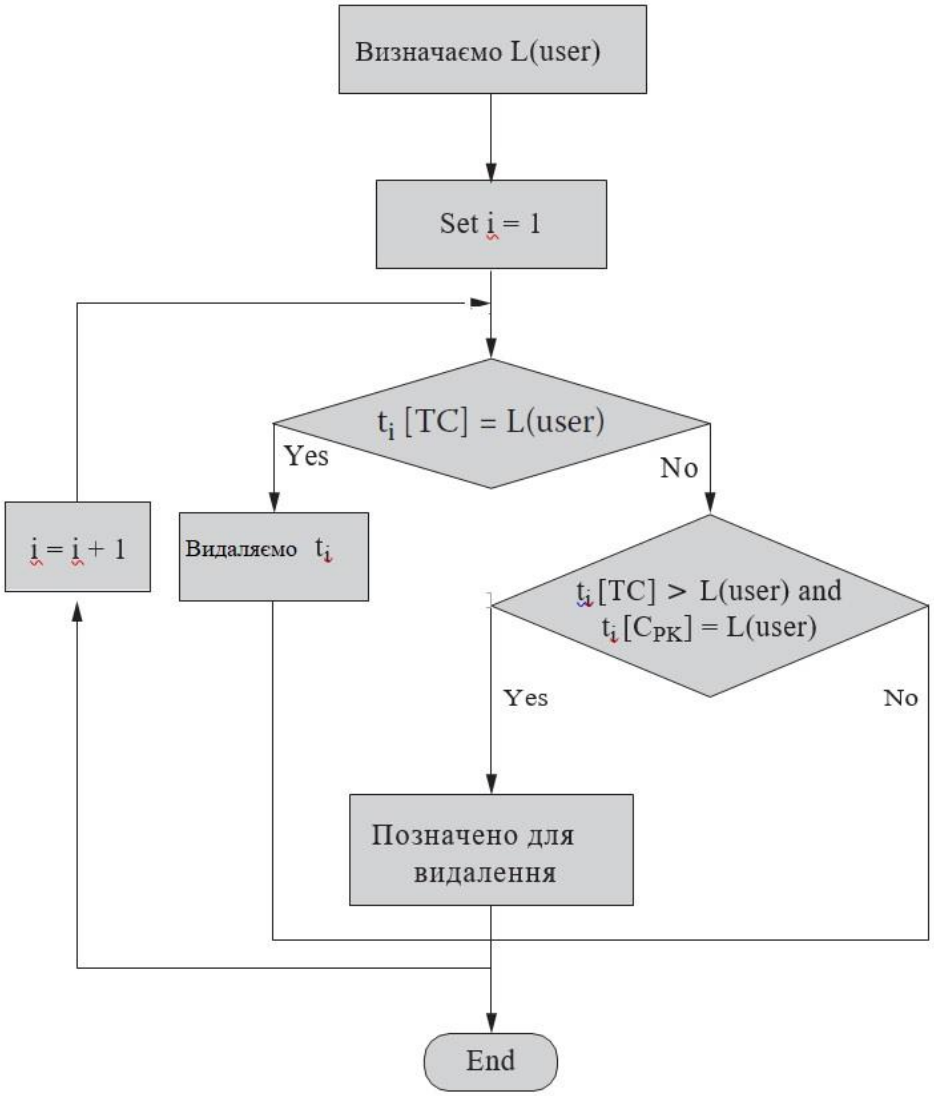

Рисунок 2.12 – Алгоритм видалення в моделі MLR

Спочатку видаляються всі записи з багаторівневого відношення, які задовольняють умові видалення і мають класифікатор безпеки запису, що дорівнює рівню користувача.

Якщо є запис, що задовольняє умові видалення і є запозиченим користувачем високого рівня(тобто рівень безпеки первинного ключа дорівнює рівню

користувача, а класифікатор безпеки запису при цьому на вищому рівні за рівень користувача), користувач високого рівня отримає повідомлення про виключення або попередження про видалення цього запозиченого запису.

2.3.5 Операція UPLEVEL

З усіх представлених моделей лише модель MLR має операцію Uplevel, що має наступний вигляд:

«UPLEVEL R

GET [A_1, A_2, ... , A_n] FROM [C_1, C_2, ... , C_n]

WHERE P»,

де R - відношення MLS, A_1, A_2, ..., A_n - атрибути з R, C_1, C_2, ... , C_n - це класифікатори безпеки A_1, A_2, ..., A_n, P - умова, що визначає записи, які підлягають підвищенню рівня безпеки.

Користувач з рівнем безпеки L виконує операцію UPLEVEL, щоб вказати, що він довіряє записам нижчого рівня. Для цього з багаторівневого відношення спершу вибирають всі записи, що задовольняють умові P та мають класифікатор безпеки запису, що не перевищує рівень користувача. Алгоритм операції підвищення рівня продемонстрований на рис. 2.13.

Далі для кожного запису r ′ з вибраних, запис r користувача будується наступним чином: значення первинного ключа запису r та його рівень безпеки встановлюються такі ж, як у первинного ключа запису r ′.

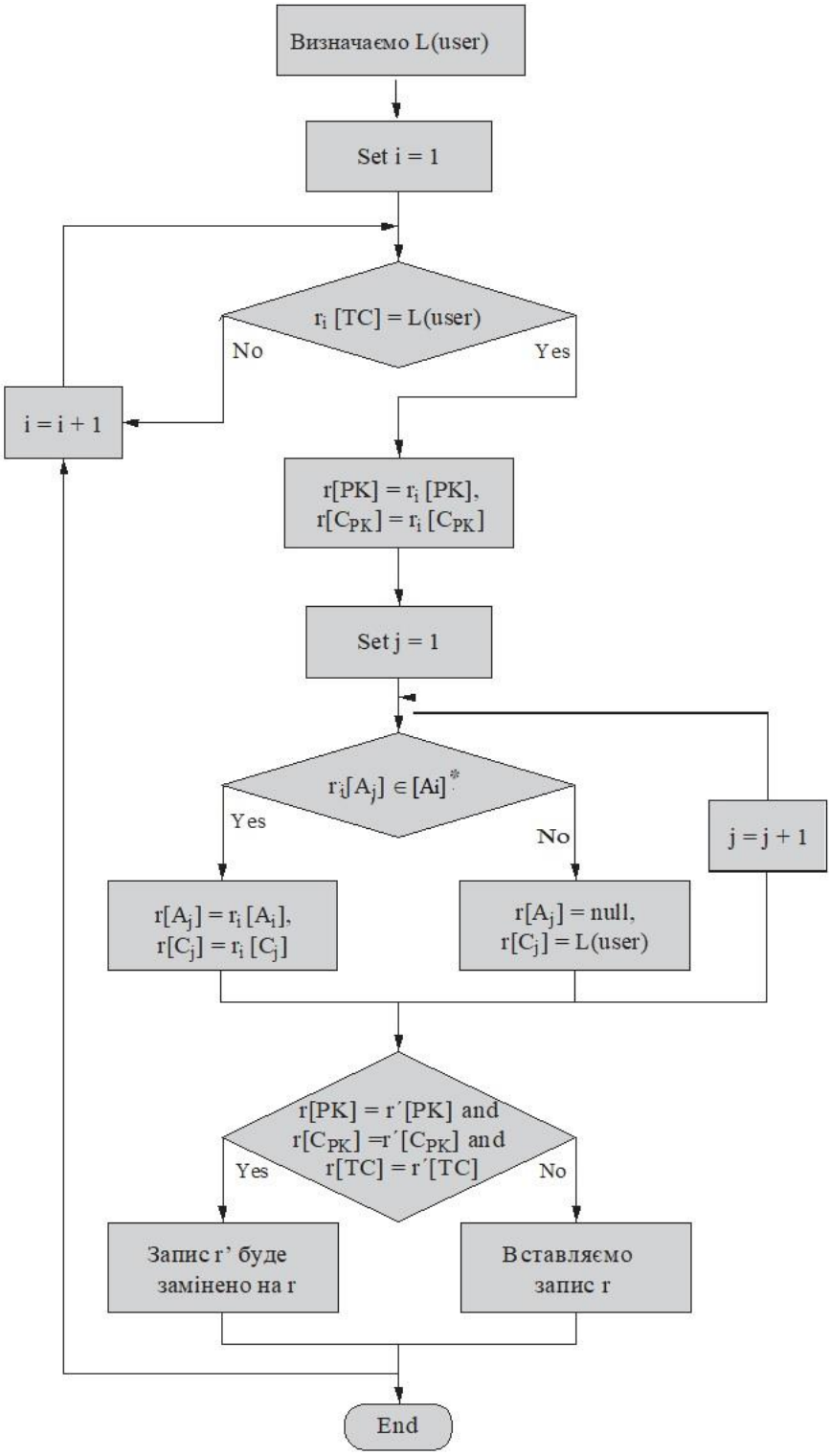

Рисунок 2.13 – Алгоритм підвищення рівня в моделі MLR

Для кожного атрибута, що не є первинним ключем та знаходиться в GET, значення атрибута запису r і його рівень безпеки будуть дорівнювати значенням цього атрибуту запису r ′ і його рівню безпеки. Якщо атрибут відсутній в GET, атрибут запису r буде дорівнювати null, а його рівень безпеки буде дорівнювати рівню безпеки користувача.

Якщо наявний запис з первинним ключем і класифікатором запису, що дорівнюють відповідно первинному ключу і класифікатору запису користувача r, цей запис буде замінено на запис r користувача. Якщо такого запису немає, у багаторівневе співвідношення вставляється запис r.

2.4 Недоліки моделей багаторівневої безпеки

Проаналізувавши наведені в роботі моделі, можна зробити припущення про недоліки та проблеми, що виникають при їх використанні. Наприклад, модель SeaView має багато проблем через алгоритми декомпозиції та відновлення. Ці проблеми можуть бути наступні:

• Повторні об’єднання:

У зв'язку з використанням вертикальної декомпозиції, в моделі SeaView запит, що включає кілька атрибутів, використовуватиме багато повторних лівих зовнішніх об'єднань між кількома однорівневими відношеннями, щоб отримати результуючий набір даних.

• Паразитні записи:

Коли алгоритм відновлення SeaView застосовується до однорівневих відношень, додаткові записи вставляються у вихідне відношення. Ці додаткові записи називаються паразитними записами і є результатом повторних об'єднань між однорівневими відношеннями.

• Неповнота:

Алгоритм декомпозиції SeaView встановлює обмеження на дані в базі. Тому у моделі SeaView неможливо реалізувати деякі відношення, які мають реальні та корисні інтерпретації в реальному житті.

Основні проблеми моделі Jajodia–Sandhu:

• Семантична неоднозначність:

Припустимо, що у відношеннях із рівнями безпеки U та S існують 2 записи, і немає запису з рівнем безпеки TS. Якщо користувач з рівнем безпеки TS повинен отримати інформацію з відношення, він не може вирішити, яка є правильною інформацією, оскільки в результаті запиту будуть отримані різні значення - із записів з рівнем U та з рівнем S.

• Оперативна неповнота:

Припустимо, що існують два непорівняні рівні безпеки, M1 і M2, чия мінімальна верхня межа - це рівень безпеки S, а найбільша нижня межа - рівень безпеки U. Користувач на рівні безпеки S не зможе вставити записи, які містять атрибути з рівнями безпеки на U, M1 і M2.

Модель Smith–Winslett також відома як модель семантики на основі переконання, але в ній додатково введена концепція базового запису. Завдяки цьому операція оновлення усуває проблеми, наявні в моделі Jajodia–Sandhu, але при цьому обмежує область оновлення до однієї сутності.

Висновки до розділу 2

В даному розділі було проведено огляд систем багаторівневої безпеки реляційних баз даних та використання декомпозиції в цих системах, пояснено використання багатозначності в реляційних базах даних з багаторівневою безпекою, а також визначено, як може виникати багатозначність і яка вона може бути. Проаналізувавши можливості та недоліки моделей, можна зробити висновок, що модель SeaView не підходить для впровадження сучасних систем в реальному житті, а модель Jajodia–Sandhu небажано використовувати через семантичну неоднозначність та неповноту, пов’язану із порівнянням рівнів безпеки.

Використовуючи алгоритми для операцій із застосуванням розмежування доступу до об’єктів БД, наведені в цьому розділі, можна впровадити ці моделі та розглянути доцільність їх впровадження в сучасних системах, що й буде описано в наступному розділі.

3 МОДЕЛЮВАННЯ СИСТЕМ З БАГАТОРІВНЕВИМ ЗАХИСТОМ БД

3.1 Використані інструменти

Для моделювання систем з багаторівневим захистом БД був створений проект, для якого використовуються Microsoft SQL Server 2017 Developer Edition [15] та Microsoft Visual Studio 2017 [16].

Сервер SQL є системою управління реляційними базами даних (RDBMS) від Microsoft, яка призначена для корпоративного середовища. SQL Server працює на T-SQL (Transact-SQL), набір розширень програмування від Sybase і Microsoft, які додають декілька функцій до стандартної SQL, включаючи керування транзакціями, виключення і обробку помилок, обробку рядків і оголошені змінні.

Microsoft Visual Studio є інтегрованим середовищем розробки (IDE) від Microsoft, що використовується для розробки консольних та графічних програм інтерфейсу користувача. Мова C# - це проста, сучасна, універсальна, об'єктно-орієнтована мова програмування.

3.2 Складові проекту

Створений проект містить базу даних, що складається з реалізованих моделей багаторівневої безпеки – SeaView, Jajodia-Sandhu, Multilevel Relational та Smith-Winslett, а також форми логіна та запиту до бази даних. Для забезпечення належного рівня захисту БД взаємодія з базою даних для звичайних користувачів здійснюється лише через спеціальні форми для взаємодії. Для входу використовуються облікові дані для бази даних. Необхідно вибрати SQL сервер

та ввести ім'я користувача і пароль(Рис. 3.1). При натисканні на кнопку «Connect» користувач проходить автентифікацію та може надсилати запити до БД.

Після успішного входу користувач отримає форму запиту (Рис. 3.2), що дозволяє одночасно створити декілька запитів та імітує цим контроль паралельності в багаторівневій безпеці бази даних, коли паралельно оброблюється декілька запитів. Ця форма містить наступні складові:

• Кнопка виконання: використовується для виконання SQL-запиту і моделювання управління паралелізмом у багаторівневій безпеці бази даних;

• Перемикач: використовується для вибору моделей багаторівневої моделі безпеки бази даних;

• Текстове поле: використовується для написання запиту,

• Data grid: використовується для перегляду результату виконання запиту до бази.

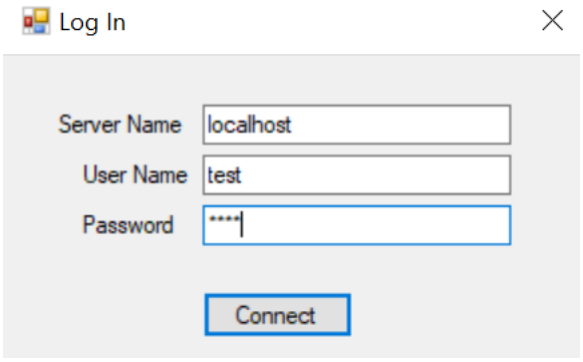

Рисунок 3.1 – Форма логіну

Для впровадження системи необхідно:

- Створити базу даних, в якій будуть окремі таблиці для кожної з представлених моделей;
- Створити ролі, які визначають рівні класифікації безпеки користувачів у багаторівневій реляційній базі даних;
- Налаштувати дозволи для ролей та користувачів;
- Визначити представлення для кожної моделі багаторівневої безпеки реляційних баз даних, з яких буде надаватись інформація для перегляду користувачам.

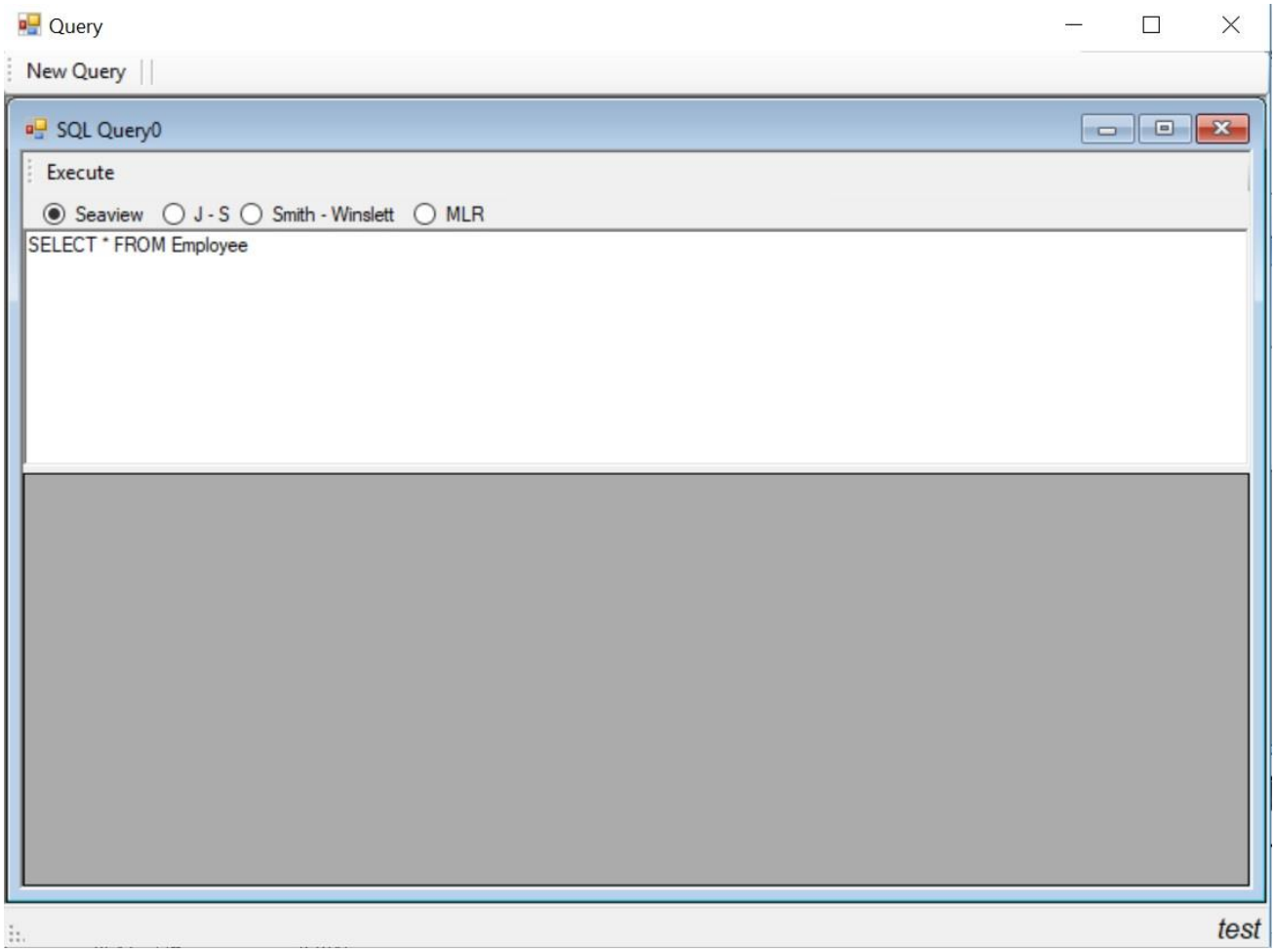

Рисунок 3.2 – Форма для взаємодії з базою даних

3.3 Дослідження продуктивності систем

Для порівняння моделей багаторівневої безпеки БД було проведене дослідження продуктивності систем. Для цього було створено базу даних, яка складається з 4 відношень. Кожна з моделей – SeaView, Jajodia-Sandhu, Smith-Winslett та MLR – була впроваджена для даної БД. Дослідження продуктивності проводилось в 3 етапи: дослідження впливу на продуктивність кількості записів у відношенні, кількості атрибутів відношення та кількості рівнів безпеки.

3.3.1 Вплив зміни кількості записів

Експеримент був розроблений для визначення того, як витрати на обробку різної кількості записів впливають на продуктивність багаторівневих моделей баз даних. Даний етап є базовим, оскільки розмір бази даних базується на кількості записів, а кількість записів, що оброблюються під час кожної транзакції, є фактором для визначення того, скільки часу потрібно для повернення відповіді користувачеві.

Дослідження проводилось з різною кількістю наповнення БД. Результати даного етапу експерименту продемонстровані на рис. 3.3.

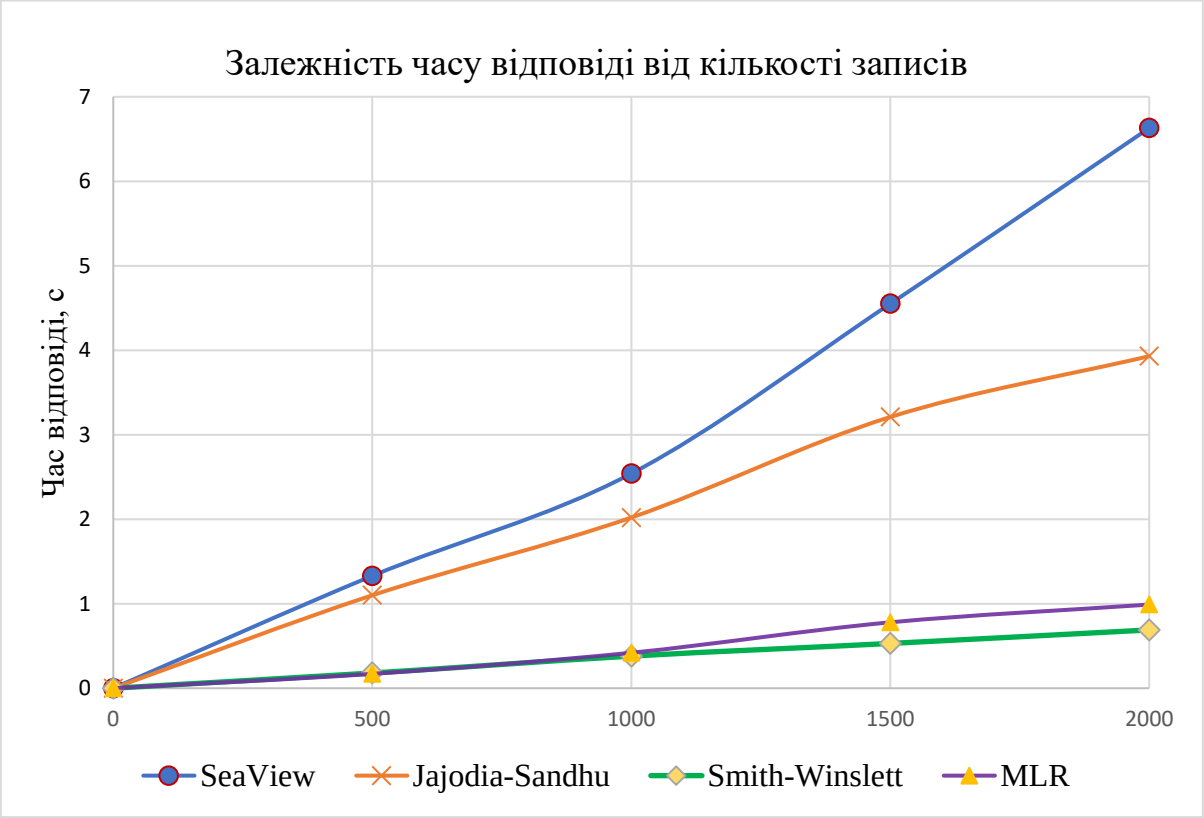

Рисунок 3.3 – Графік залежності часу відповіді від кількості записів

3.3.2 Вплив зміни кількості атрибутів відношення

Даний етап експерименту був проведений для визначення того, чи впливають і як витрати на обробку різної кількості атрибутів у відношенні на продуктивність багаторівневих моделей баз даних. Як і на попередньому етапі, дослідження проводилось з вимірами часу відповіді на запит. Результати даного етапу експерименту продемонстровані на рис. 3.4.

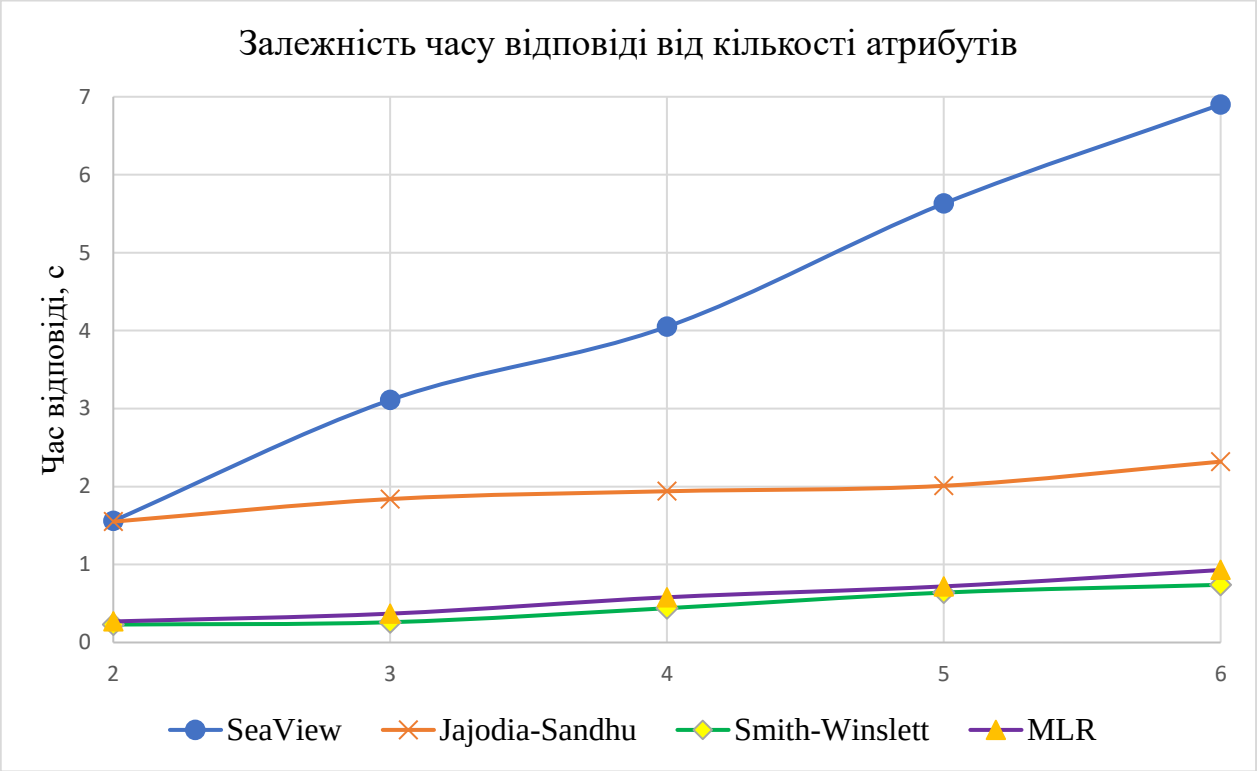

Рисунок 3.4 – Графік залежності часу відповіді від кількості атрибутів

3.3.3 Вплив зміни кількості рівнів безпеки

Останній етап експерименту був проведений для визначення того, чи впливають і як витрати на обробку різної кількості визначених рівнів безпеки на продуктивність багаторівневих моделей баз даних. Як і на попередніх етапах, дослідження проводилось з вимірами часу відповіді на запит. Результати експерименту продемонстровані на рис. 3.5.

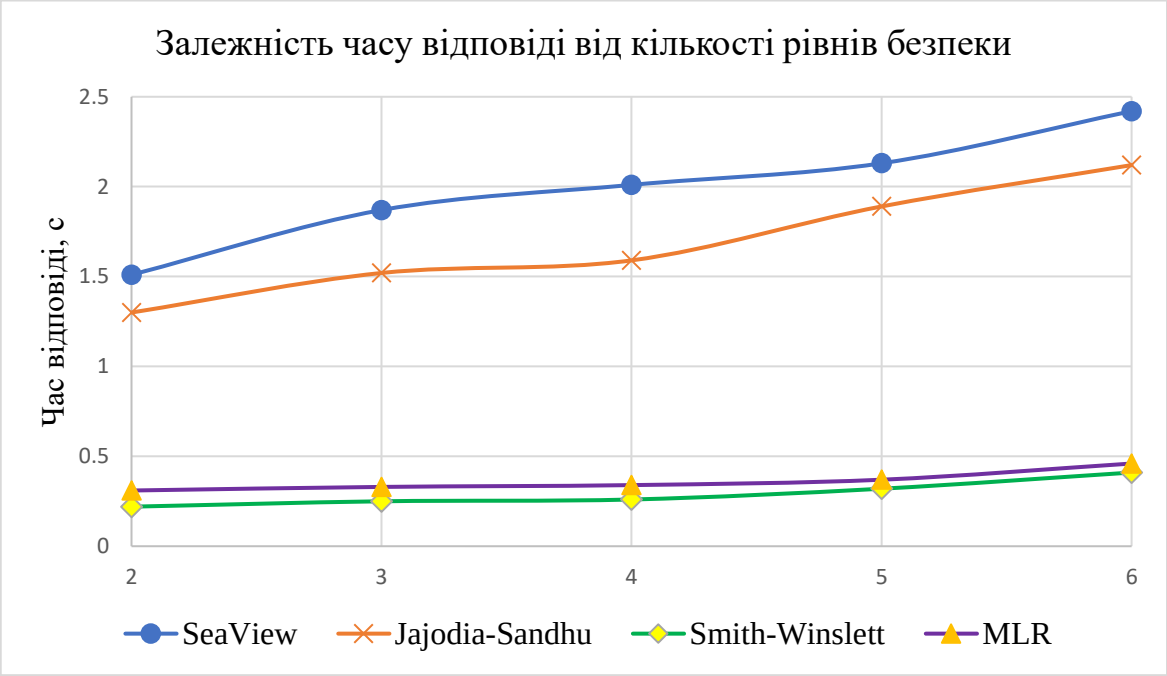

Рисунок 3.5 – Графік залежності часу відповіді від кількості рівнів безпеки

Висновки до розділу 3

В даному розділі було представлене дослідження із застосуванням змодельованих систем з багаторівневою безпекою БД. Результати порівняльного дослідження дають змогу пояснити сильні та слабкі сторони кожної моделі.

Очевидна перевага моделей MLR та Smith-Winslett пов’язана з тим, що дані в них не піддавались декомпозиції, тож не зберігаються у вигляді однорівневих відношень. Продуктивність моделі Smith-Winslett є найвищою, оскільки вона не підтримує класифікацію безпеки на рівні кожного окремого атрибута; рівні доступу можуть бути призначені тільки для ключових атрибутів і для записів в цілому. Модель MLR пропонує меншу продуктивність, ніж модель Smith-Winslett, оскільки вона підтримує класифікацію на рівні кожного окремого

атрибута. Модель Jajodia–Sandhu працює набагато гірше за моделі MLR та Smith-Winslett через вплив операції об'єднання між однорівневими відношеннями в алгоритмі відновлення. Модель SeaView є найменш ефективною з усіх через вплив операцій з'єднання між вертикальними однорівневими відношеннями та об'єднання між горизонтальними однорівневими відношеннями в алгоритмі відновлення.

ВИСНОВКИ

Безпечне керування конфіденційними даними є критичним питанням в сучасних організаціях та державних установах, тому для підвищення захищеності баз даних необхідно впроваджувати системи з багаторівневою безпекою. В ході даної роботи було розглянуто та проаналізовано концепцію багаторівневої безпеки баз даних, встановлені переваги та недоліки використання різних типів управління доступом в реляційних БД. Були розроблені алгоритми реалізації операцій реляційної алгебри в системах з багаторівневою безпекою і на їх основі побудований програмний комплекс моделювання баз даних з багаторівневою безпекою.

За результатами проведеного дослідження було встановлено, що кращими за показниками, як і очікувалось, є моделі MLR та Smith-Winslett. Вони є приблизно однаковими за продуктивністю, тож варто вибирати між ними, виходячи з даних, якими оперує підприємство, та недоліків кожної з моделей. Модель Jajodia–Sandhu не є ефективною, а також не є зручною для використання на великих об’ємах інформації. Стосовно моделі SeaView можна сказати, що вона морально застаріла і несумісна з використанням в сучасному житті, бо дає жахливі результати в порівнянні з іншими моделями.

Отже, моделі Multilevel Relational та Smith-Winslett є простими у впровадженні, захищеними, однозначними і досить потужними для реалізації сучасних систем із багаторівневою безпекою, в тому числі і для використання державними установами України. Таким чином, запропонований підхід до моделювання систем багаторівневої безпеки показав свою ефективність, а його практична реалізація є кінцевим рішенням.

СПИСОК ДЖЕРЕЛ ПОСИЛАНЬ

1. Library of free on-line IT books [Електронний ресурс] – Режим доступу до ресурсу: https://www.techotopia.com.

2. SearchSecurity [Електронний ресурс] – Режим доступу до ресурсу: https://searchsecurity.techtarget.com/.

3. Connolly T. Database Systems. A Practical Approach to Design, Implementation, and Management [Текст] / T. Connolly, C. Begg, 2015. – (6), 168-180.

4. Hellerstein J.M. Architecture of a database system. Foundations and Trends in Databases [Текст]: J. M. Hellerstein, Michael Stonebraker, James Hamilton, 2007. – 141–259 с.

5. Bertino E. Database security - Concepts, approaches, and challenges. IEEE Transactions on Dependable and Secure Computing 2 [Текст] / E. Bertino, R. Sandhu, 2005. – 2–19.

6. Pierangela S. Access control: Policies, models, and mechanisms. In Foundations of security analysis and design [Текст] / S. Pierangela, D. Sabrina. – Berlin: Springer, 2001. – 137–196.

7. Walid R. A multi-purpose implementation of mandatory access control in relational database management systems [Текст] / R. Walid, P. Bird. – Toronto, 2004. – 1010–1020.

8. Giuri L. A role-based secure database design tool [Текст] / L. Giuri, P. lglio. – Proceedings of the 12th Annual Computer Security Applications Conference, 1996. – 203–212.

9. Документація IBM [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/support/knowledgecenter.

10. Xu L. Study on methods for data confidentiality and data integrity in relational databases [Текст]: L. Xu, D. Sun, D. Liu. – ICCSIT, 2010. – 292–295.

11. Mikko T.S. Database security and the problem of polyinstantiation: A moral scrutiny [Текст] / T.S. Mikko. – Australian Journal of Information Systems 10 (1), 2002. – 41–49.

12. Nelson D. Using polyinstantiation to develop an MLS application [Текст] / D. Nelson, C. Paradise. – Computer Security Applications Conference, 1991. – 12–22.

13. Multilevel security for relational databases [Текст] / S. Osama, M. El-Sayed, S. Hala та ін. – CRC Press, 2014. – 35–53.

14. Стаття з журналу [Електронний ресурс] – Режим доступу до ресурсу: https://profsandhu.com/journals/tissec/t98mlr.pdf.

15. Документація Microsoft SQL Server [Електронний ресурс] – Режим доступу до ресурса: https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation.

16. Документація Microsoft Visual Studio [Електронний ресурс] – Режим доступу до ресурса: https://docs.microsoft.com/en-us/visualstudio/.

ДОДАТКИ

Додаток А

1) Клас globals

```
using System;
using System.Configuration;
using System.Collections;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace GlobalsClass
{
    class globals
    {
        public static string ServerName;
        public static string Password;
        public static string UserName;
        public static string ServerConnStr;
        public static string UserLabel;
        public static int UserLabelID;

        public static string Scrub(string text)
        {
                return text.Replace(" ", "");
        }

        public static string[] KeyWords = new string[8];

        public enum SqlStatment
        {
            select, Insert, update, Delete
             }
```

2) Клас DBOperations

```
namespace GlobalsClass.DBOperations
{
    public class DBOperations
    {
        public DBOperations()
        {
        }


        public static void SqlConn(string Server, string User, string Pass)
        {
            globals.ServerConnStr = "Data Source = " + Server + "; Initial Catalog = MLSDB;
User Id = " +User + "; Password = " +Pass + ";";
        }

        public static DataSet GetData(string SqlStr)
        {
            DataSet ds = new DataSet();
            string SqlConnStr = globals.ServerConnStr;
            SqlConnection SqlConn = new SqlConnection(SqlConnStr);
            SqlConn.Open();
            SqlCommand SqlCmd = new SqlCommand(SqlStr, SqlConn);
            SqlDataAdapter Adpt = new SqlDataAdapter(SqlCmd);
            Adpt.Fill(ds);
            SqlConn.Close();
            return ds;
        }

        public static void FillDataSet(ref DataSet DS, string DT, string SqlStr)
        {
           string SqlConnStr = globals.ServerConnStr;
           SqlConnection SqlConn = new SqlConnection(SqlConnStr);
           SqlConn.Open();
           SqlCommand SqlCmd = new SqlCommand(SqlStr, SqlConn);
           SqlDataAdapter Adpt = new SqlDataAdapter(SqlCmd);
           Adpt.Fill(DS, DT);
           SqlConn.Close();
        }

        public static string DateFormate(string Date)
        {
            string[] dateMDY = Date.Split('/');
            string DateDMY = dateMDY[1] + '/' +dateMDY[0] + '/' +dateMDY[2]; return DateDMY;
        }

        public static int? StringToNullableInt32(string s)
        {
            int i;
            if (Int32.TryParse(s, out i)) return i;
            return null;
```

```
    }

    public static void SetData(string SqlStr)
    {
        string SqlConnStr = globals.ServerConnStr;
        SqlConnection SqlConn = new SqlConnection(SqlConnStr);
        SqlConn.Open();
        SqlCommand SqlCmd = new SqlCommand(SqlStr, SqlConn);
        SqlCmd.ExecuteNonQuery();
        SqlConn.Close();
    }
  }
}
```

3) Клас MLSDB

```
namespace GlobalClasses
{
    class MLSDB
    {
        public static string DMLSTR(String SQLSTR)
        {

            string DML = "";
            if (SQLSTR.ToUpper().Contains("SELECT"))
            {
                if (SQLSTR.ToUpper().Contains("WHERE"))
                {
                    DML = SQLSTR.Substring(SQLSTR.ToUpper().IndexOf("FROM") + 4,
SQLSTR.ToUpper().IndexOf("WHERE") + (SQLSTR.ToUpper().IndexOf("FROM") + 4));
                }
                else
                {
```

```
                        DML = SQLSTR.Substring(SQLSTR.
                        ToUpper().IndexOf("FROM") + 4);
                    }
                }
                else if (SQLSTR.ToUpper().Contains("UPDATE"))
                {
                    DML = SQLSTR.Substring(SQLSTR.ToUpper().IndexOf("UPDATE") + 6,
SQLSTR.ToUpper().IndexOf("SET") - (SQLSTR.ToUpper().IndexOf("UPDATE") + 6));
                }
                else if (SQLSTR.ToUpper().Contains("INSERT"))
                {
                    DML = SQLSTR.Substring(SQLSTR.ToUpper().IndexOf("INSERT") + 6,
SQLSTR.ToUpper().IndexOf("VALUES") - (SQLSTR.ToUpper().IndexOf("INSERT") + 6));


                }
                else if  (SQLSTR.ToUpper().Contains("UPLEVEL"))
     {
                    DML = SQLSTR.Substring(SQLSTR.ToUpper().
                    IndexOf("UPLEVEL") + 7, SQLSTR.
                    ToUpper().IndexOf("GET") - (SQLSTR.ToUpper().IndexOf("UPLEVEL") + 7));
                }
                else if (SQLSTR.ToUpper().Contains("VERIFY"))
                            {
                    if (SQLSTR.ToUpper().Contains("TRUE"))
                    {
                        DML = SQLSTR.Substring(SQLSTR.ToUpper().IndexOf("TRUE") + 4,
SQLSTR.



                        ToUpper().IndexOf("WHERE") - (SQLSTR.ToUpper().IndexOf("TRUE") +
4));
                    }
                    else if (SQLSTR.ToUpper().
```

```
                    Contains("FALSE"))
                    {
                        DML = SQLSTR.Substring(SQLSTR.ToUpper().
                        IndexOf("FALSE") + 5, SQLSTR.ToUpper().IndexOf("WHERE") - (SQLSTR.
                        ToUpper().IndexOf("FALSE") + 5));
                    }
                }
                else if (SQLSTR.ToUpper().Contains("DELETE"))
                {
                    if (SQLSTR.ToUpper().Contains("WHERE"))
                    {
                        DML = SQLSTR.Substring(SQLSTR.ToUpper().
                        IndexOf("FROM") + 4, SQLSTR.ToUpper().IndexOf("WHERE") - (SQLSTR.
                        ToUpper().IndexOf("FROM") + 4));
                    }
                    else
                    {
                        DML = SQLSTR.Substring(SQLSTR.
                        ToUpper().IndexOf("FROM") + 4);
                    }
                }
                return DML;
            }

            public static string AttributeSTR(String SQLSTR)
            {
                string DML = "";

                if (SQLSTR.ToUpper().Contains("SELECT"))
                {
                    DML = SQLSTR.Substring(SQLSTR.ToUpper().IndexOf("SELECT") + 6,
SQLSTR.ToUpper().IndexOf("FROM")
```

```
                    - (SQLSTR.ToUpper().IndexOf("SELECT") + 6));
                }
                else if (SQLSTR.ToUpper().Contains("UPDATE"))
                {
                    if (SQLSTR.ToUpper().Contains("WHERE"))

                    {
                        DML = SQLSTR.Substring(SQLSTR.ToUpper().IndexOf("SET") + 3,
SQLSTR.ToUpper().IndexOf("WHERE") - (SQLSTR.ToUpper().IndexOf("SET") + 3));
                    }
                    else
                    {
                        DML = SQLSTR.Substring(SQLSTR.ToUpper().IndexOf("SET") + 3);
                    }
                }
                else if (SQLSTR.ToUpper().Contains("UPLEVEL"))
                {
                    if (SQLSTR.ToUpper().Contains("WHERE"))
                    {
                        DML = SQLSTR.Substring(SQLSTR.ToUpper().IndexOf("GET") + 3,
SQLSTR.ToUpper().IndexOf("WHERE") - (SQLSTR.ToUpper().IndexOf("GET") + 3));
                    }
                    else
                    {
                        DML = SQLSTR.Substring(SQLSTR.ToUpper().IndexOf("SET") + 3);
                    }
                }
                else if (SQLSTR.ToUpper().Contains("VERIFY"))
                {
                    if (SQLSTR.ToUpper().Contains("TRUE"))
                    {
                        DML = "TRUE";
```

```
            }
            else
            {
                DML = "FALSE";
            }
        }
        else if (SQLSTR.ToUpper().Contains("INSERT"))
        {
            DML = SQLSTR.Substring(SQLSTR.
            ToUpper().IndexOf("VALUES") + 6);
        }
        else if (SQLSTR.ToUpper().Contains("DELETE"))
        {
            if (SQLSTR.ToUpper().Contains("WHERE"))
            {
                DML = SQLSTR.Substring(SQLSTR.
                ToUpper().IndexOf("WHERE") + 5);
            }
            else
            {
                DML = "";
            }
        }

        return DML;
    }
    public static string PredicateSTR(String SQLSTR)
    {
        string DML = "";
        if (SQLSTR.ToUpper().Contains("WHERE"))
        {
            DML = SQLSTR.Substring(SQLSTR.ToUpper().IndexOf("WHERE") + 5);
```

```
                }
                else
                {
                    DML = "";
                }
                return DML;
            }
            public static int GetUserLabelID()
            {
                int UserLabel = 0;
                string SqlStr = "select[dbo].GetLabelID(select[dbo].[GetUserLabel]())";
                UserLabel = 2;
//int.Parse(DBOperations.GetData(SqlStr).Tables[0].Rows[0][0].ToString());
                return UserLabel;
            }
            public static string GetUserLabel()
            {
                string UserLabel = "";
                string SqlStr = "select[dbo].[GetUserLabel]() ";
                UserLabel =
"TS";//DBOperations.GetData(SqlStr).Tables[0].Rows[0][0].ToString();
                return UserLabel;
            }


            public static int GetLabelID(string UserLabel)
            {
                int UserLabelID = 0;
                string SqlStr = "select[dbo].GetLabelID('" + UserLabel + "') ";
                UserLabelID =
int.Parse(DBOperations.GetData(SqlStr).Tables[0].Rows[0][0].ToString());
                return UserLabelID;
            }
```

```
            public static string GetBCLabel(int UserLabelID)
            {
                string UserLabel = ""; string UChar = ""; string CChar = ""; string SChar =
"; string TSChar = "";
                UChar = (UserLabelID % 4).ToString(); UserLabelID = UserLabelID / 4;
                CChar = (UserLabelID % 4).ToString(); UserLabelID = UserLabelID / 4;
                SChar = (UserLabelID % 4).ToString(); UserLabelID = UserLabelID / 4;
                TSChar = (UserLabelID % 4).ToString(); if (UChar == "1")
                {
                    UserLabel = UserLabel + "-U";
                }
                else if (UChar == "2")
                {
                    UserLabel = UserLabel + "U" ;
                }
                if (CChar == "1")
                {
                    UserLabel = UserLabel + "-C" ;
                }
                else if (CChar == "2")
                {
                    UserLabel = UserLabel + "C";
                }
                if (SChar == "1")
                {
                    UserLabel = UserLabel + "-S";
                }
                else if (SChar == "2")
                {
                    UserLabel = UserLabel + "S";
                }
                if (TSChar == "1")
```

```
                {
                   UserLabel = UserLabel + "-T";
                }
                else if (TSChar == "2")
                {
                    UserLabel = UserLabel + "T";
                }

                return UserLabel;
            }



            public static int GetBCLabelNumeric(string UserLabel)
            {
                int UserLabelID = 0;
                UserLabel = BreakBCLabel(UserLabel);
                UserLabelID = int.Parse(UserLabel.Substring(0, 1)) +
(int.Parse(UserLabel.Substring
                (1, 1)) * 4) + (int.Parse(UserLabel.
                Substring(2, 1)) * 16) + (int.
                Parse(UserLabel.Substring(3, 1)) * 64); return UserLabelID;
            }
            public static string GetBCUserView(string Label)
            {
                string labelView = ""; int NumericLabel = 0;
                Label = BreakBCLabel(Label);
                int UserLabelID = globals.UserLabelID; for (int i = 0; i < UserLabelID;
i++)
                {
                    NumericLabel = NumericLabel + (int.Parse(Label.Substring(i, 1)) *
Convert.ToInt32(Math.Pow(4, i)));
                }
                labelView = GetBCLabel(NumericLabel); return labelView;
```

```
    }
public static string BreakBCLabel(string UserLabel)
{
    string UserLabelID = "0";
    string Unumric = "0";
    string Cnumric = "0";
    string Snumric = "0";
    string TSnumric = "0";

    if (UserLabel.Contains('U'))
    {
            if (UserLabel.IndexOf('U') == 0)
            {
                Unumric = "2";
            }
            else
            {
                if (UserLabel.Substring(UserLabel.IndexOf('U') - 1, 1) == "-")
                {
                    Unumric = "1";
                }
                else
                {
                    Unumric = "2";
                }
            }
    }

    if (UserLabel.Contains('C'))
    {
            if (UserLabel.IndexOf('C') == 0)
            {
```

```
            Cnumric = "2";
        }
        else
        {
            if (UserLabel.Substring(UserLabel.IndexOf('C') - 1, 1) == "-")
            {
                Cnumric = "1";
            }
            else
            {
                Cnumric = "2";
            }
        }
}

if (UserLabel.Contains('S'))
{
    if (UserLabel.IndexOf('S') == 0)
    {
        Snumric = "2";
    }
    else
    {
        if (UserLabel.Substring(UserLabel.IndexOf('S') - 1, 1) == "-")

        {
            Snumric = "1";
        }
        else
        {
            Snumric = "2";
        }
```

```
        }
    }
    if (UserLabel.Contains('T'))
    {
        if (UserLabel.IndexOf('T') == 0)
        {
            TSnumric = "2";
        }
        else
        {
            if (UserLabel.Substring(UserLabel.IndexOf('T') - 1, 1)   == "-")
            {
                    TSnumric = "1";
            }
            else
            {
                    TSnumric = "2";
            }
        }
    }

    UserLabelID = Unumric + Cnumric + Snumric + TSnumric;
    return UserLabelID;
}

public static string GetBCprimarylevel(int NumericLabel)
{
    string Label = GetBCLabel(NumericLabel);
    return Label.Substring(0, 1);
}

public static string GetBCSecondarylevel(int NumericLabel)
```

```
            {
                string Label = GetBCLabel(NumericLabel); return
Label.Remove(Label.IndexOf(GetBCprimarylevel(NumericLabel)), 1);
            }

            public static string GetBCUserbelief(string Label)
            {
                int NumericLabel = GetBCLabelNumeric(Label); int belief = 0;
                string retvalue = "";

                int UserLabelID = globals.UserLabelID;
                string UserLabel = globals.UserLabel;

                for (int i = 0; i < UserLabelID; i++)
                {
                    belief = NumericLabel % 4;
                    NumericLabel = NumericLabel / 4;
                }

                if (belief == 1)
                {
                    retvalue = "-" + UserLabel;
                }
                else if (belief == 2)
                {
                    retvalue = UserLabel;
                }

                return retvalue;
            }

            public static int UnverifyBCUserbelief(int NumericLabel)
```

```
{
    string Label = GetBCLabel(NumericLabel);
    string UserLabel = globals.UserLabel;
    Label = Label.Remove(Label.IndexOf(UserLabel), 1);

    return GetBCLabelNumeric(Label);
}

public static int VerifyBCUserbelief(int NumericLabel, bool belief)
{
    string Label = GetBCLabel(NumericLabel);

    string UserLabel = globals.UserLabel;
    int UserLabelID = globals.UserLabelID;

    if (belief)
    {
        Label = Label.Insert(UserLabelID - 1, globals.UserLabel);
    }
    else
    {
        Label = Label.Insert(UserLabelID - 1,"-"+globals.UserLabel);
    }

    return GetBCLabelNumeric(Label);
}

public static ArrayList GetAttribute(string Str)
{
    //ArrayList ReturnARR = new ArrayList();
    ArrayList ARR = new ArrayList();
```

```
            string Attribute = "";
            string Values = "";

            if (Str.Contains(','))
            {
                ARR.AddRange(Str.Trim().Split(','));
            }
            else
            {
                ARR.Add(Str.Trim());
            }

            return ARR;
        }

        public static ArrayList GetInsertValue(string Str)
        {
            ArrayList ARR = new ArrayList();

            Str = Str.Remove(Str.IndexOf('('), 1);
            Str = Str.Remove(Str.IndexOf(')'), 1);

            if (Str.Contains(','))
            {
                ARR.AddRange(Str.Trim().Split(','));
            }

            return ARR;
        }

    }
}
```