КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (23), 2024
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2024.23.4255
УДК 004.65

Момрик Ярослава Богданівна
асистент кафедри захисту інформації
Національний Університет «Львівська політехніка», Львів, Україна
ORCID 0009-0001-4114-0347
sm.slyala@gmail.com

Ящук Юрій Олександрович
кандидат фізико-математичних наук, доцент, завідувач кафедри прикладної математики
Львівський національний університет ім. І. Франка, Львів, Україна
ORCID ID 0000-0003-4935-497X
yuriy.yashchuk@lnu.edu.ua

Тучапський Роман Ігорович
кандидат фізико-математичних наук, старший науковий співробітник
Інститут прикладних проблем механіки і математики ім. Я. С. Підстригача НАН України, Львів, Україна
ORCID 0000-0002-2556-1088
roman.tuch@gmail.com

ПРОФЕСІЙНИЙ ПІДХІД ЯК МЕТОД ЗАХИСТУ ІНФОРМАЦІЇ НА ЕТАПАХ РОЗРОБКИ РЕЛЯЦІЙНИХ БАЗ ДАНИХ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОБОТИ З НИМИ

Анотація. Проаналізовано проектування реляційних Баз Даних (БД) та Програмного Забезпечення (ПЗ) для роботи з ними з точки зору складових проектування та розробки, що впливають на безпеку інформації, та від правильності виконання яких залежить безпека даних, названо внутрішні загрози, які виникають через недосконале проектування. Описано позитивні практики, що дозволяють проектувати БД та розробляти ПЗ для роботи з ними в аспекті безпечного коду. Обґрунтовано, чому етап проектування реляційних баз даних, коли застосовується нормалізація відношень, формуються зв'язки між таблицями та обмеження цілісності, є кроком по забезпеченню захисту даних, на чому не наголошується в літературі щодо безпеки баз даних, зокрема, показано як створювати зовнішні зв’язки між таблицями так, щоб Система Керування Базами Даних (СКБД) захищала дані від порушення цілісності і тд. Висвітлено моменти в подальшій розробці ПЗ, які несуть відповідальність за безпеку роботи з базою даних з точки зору створення надійного та безпечного коду, виходячи з практичного досвіду програміста. Підхід безпечного коду широко вживається при розробці ПЗ та на рівні аудиту програмного забезпечення, оскільки він дозволяє запобігати внутрішнім загрозам захисту інформації, які є найпоширенішою причиною втрати безпеки даних, вимоги до використання цього підходу включені в оновлені стандарти інформаційної безпеки, тому рекомендовано їх враховувати професійним розробникам баз даних та ПЗ.

Ключові слова: безпека коду; безпека даних; проектування баз даних, конструювання програмного забезпечення; загрози безпеки; цілісність даних; обмеження цілісності; нормалізація баз даних.

ВСТУП

Постановка проблеми. Коли застосовуємо термін «захист інформації» до бази даних, першими виникають асоціації з шифруванням даних, протоколами запитів та резервним копіюванням, паролями та логінами, правами та ролями користувачів, захистом серверів та захистом від SQL-ін’єкцій. Але безпека даних залежить від

професійності підходу на кожному кроці від проектування бази даних до розробки та впровадження ПЗ для роботи з нею, і лише при ефективному та якісному виконанні цих кроків, решта вищезгаданих засобів захисту будуть ефективними.

Мета статті. Метою цієї статті є проаналізувати кроки проектування, виділивши ті моменти, що пов'язані чи безпосередньо впливають на захист даних, та зібрати кращі практики, які дозволяють використати механізми коректного збереження інформації у реляційних БД та подальшої безпечної роботи з даними.

Для досягнення мети потрібно виконати завдання:

- Проаналізувати кроки проектування реляційних баз даних та ПЗ та виявити ті, що впливають на безпеку даних.
- Сформулювати ризики, яких можна уникнути чи критерії надійності, яких можна досягнути, здійснивши аналіз в розумінні безпечного коду.
- Для уникнення цих ризиків запропонувати методи проектування та розробки, які найбільш доцільно вживати згідно з практичним досвідом розробки та рекомендацій професійних розробників чи дослідників.
- У частині розробки ПЗ за доцільності описати оптимальні алгоритми дій для мінімізації ризиків стосовно різних предметних областей проектування та розробки.

Аналіз останніх досліджень і публікацій. Література, котра зустрічається в інтернет просторі — як статистичне відображення легко доступної інформації на дану тематику, як правило, аналізує та визначає загрози даним та дає визначення безпеки БД [5], [7], а також дає рекомендації щодо створення особливо захищених баз даних. Водночас, автори не наголошують, що вже грамотне виконання звичайних кроків проектування баз є способом забезпечення захисту даних. Серйозною загрозою безпеці даних є не лише зовнішні небезпеки, але й внутрішні загрози, викликані некоректною розробкою чи використанням. В цьому аспекті перш за все виникають наступні загрози [9]:

- загрози цілісності (знищення та модифікація інформації);
- загрози доступності (блокування та знищення інформації);
- загрози конфіденційності (Несанкціонований Доступ (НСД), витік та розголошення інформації).

Щоб захистити базу даних, необхідно захистити усі складові комплексної системи, а саме [1]:

- дані у базі даних;
- систему управління базами даних;
- всі пов'язані додатки;
- фізичний та/або віртуальний сервер бази даних та базове обладнання;
- обчислювальну та/або мережну інфраструктуру, яка використовується для доступу до БД.

У [12] подано опис основних загроз базам даним та рекомендації щодо методів усунення. Джерела літератури, такі як [2] зосереджуються на винайдені методів створення захищених баз даних або пропонують багаторівневий контроль доступу-авторизація і шифрування — Osama [10]. Одна з найновіших публікацій — рекомендації щодо застосування методики DevOps для роботи з базами даних у хмарних середовищах — йдуть пошуки особливо захищених методик та залишається ніша, яку розглянуто в [11] — йдеться про застосування методології розробки безпечного програмного забезпечення зокрема щодо «S-SDLC (Життєвого циклу розробки безпечного програмного забезпечення). Він заснований на перевірці вимог безпеки на різних етапах розробки програмного

забезпечення: аналіз, проектування, розробка, тестування та обслуговування. Особливо під час перших двох, оскільки більшість слабких місць систем генеруються ще до початку завдань програмування» — саме ця тема розкрита у статті щодо внутрішніх загроз при проектуванні реляційних баз даних, де автори мають змогу робити висновки та надавати рекомендацій з огляду на власний досвід проектування, розробки, тестування та впровадження баз даних та пз. В цьому ж напрямку працювали автори [3], вони провели дослідження саме баз даних, побудованих на основі схеми з універсальною основою відносин, що реалізується в рамках реляційного моделі даних і представили методи та засоби, які забезпечують цілісність основних компонентів таких баз даних, відштовхуючись від моделі цілісності Кларка–Вілсона [3] — «Предмет транзакції, що не порушує цілісність об'єкта». Програми — користувачі відповідно до цієї моделі не мають прямого доступу до об'єктів. Об'єкти можуть бути тільки доступ через процедуру перетворення (ТР) [3] — це напрям для побудови особливо захищених баз даних та додатків, де контроль цілісності особливо важливий, бо ризики помилок виникають при реалізації концепції моделі. У нашому ж матеріалі охоплено більш глобально проблему з точки зору традиційної побудови реляційних баз.

Надійне проектування стуктури БД та ПЗ є першою потебою а його відсутність першою загрозою, про яку говорить [1] — «Внутрішні загрози є найпоширенішими причинами порушень безпеки БД».

Отже на часі висвітлити ті причини загроз, які часто закладаються вже на етапі проектування баз даних та додатків для роботи з ними. Маркус Єтелька [6] пише: «Висока важливість безпечного кодування підкреслюється в оновлених стандартах інформаційної безпеки ISO/IEC 27002 та ISO/IEC 27001:2022. Як новий захід (контроль) інформаційної безпеки, принципи «Безпечного кодування» при розробці програмного забезпечення вносяться до каталогу заходів у Додатку А стандарту ISO/IEC 27001». Тому ця стаття покликана бути своєчасним відгуком на оновлення стандартів, щоб знаходити та відслідковувати всі ланки які відповідають за безпеку даних, а не лише ті, котрі прямо асоціюються в літературі з засобами безпеки.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

1. Проектування баз даних. Поняття захисту даних, що зберігаються у базі даних, споріднене з терміном «захист інформації» (англ. Data protection). За даними Вікіпедії — це сукупність методів і засобів, що забезпечують цілісність, конфіденційність і доступність інформації за умов впливу на неї загроз природного або штучного характеру, реалізація яких може призвести до завдання шкоди власникам і користувачам інформації.

Тому першим етапом захисту даних на етапі розробки є правильне проектування бази даних — яке забезпечить дотримання цілісності даних та захист від втрати їх актуальності при роботі з базою. Тут основними аспектами є нормалізація та правила цілісності закладенні при проектуванні структури, зокрема типи полів, які є ключовими та полів, що слугують для зовнішніх зв'язків між таблицями (захист від аномалій вилучення, вставки та корегування) — схеми та зв'язків між таблицями у випадку реляційної моделі як наведено на рис. 1.

ISSN 2663 - 4023

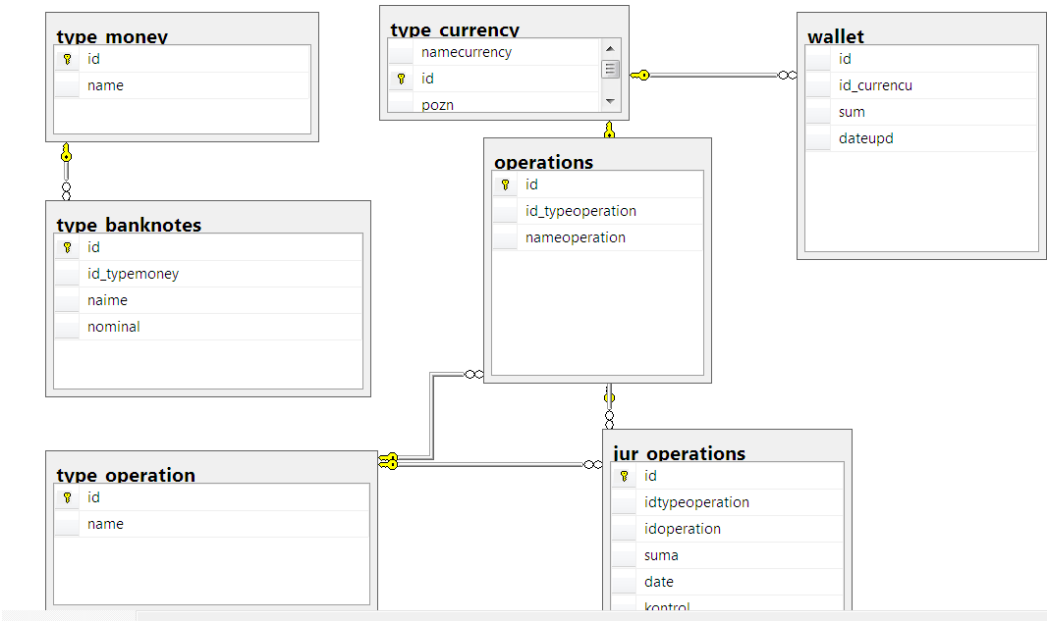

Рис. 1. Схема реляційної бази даних

2. Підвалини захисту на рівні СКБД. На етапі проектування бази даних враховуємо: цілісність відношень, цілісність атрибутів, цілісність зв'язків між відношеннями, що відображається у схемі бази даних. Це означає, що для створення (власне якщо цього не зробити, то цілісність сутностей буде порушена, на основі визначення цього терміну) на етапі проектування треба правильно підбирати тип ключового поля враховуючи, що воно має бути ненульове та унікальне (інакше ключ не створиться) і має мати тип, зручний для побудови зовнішніх зв'язків з дочірніми таблицями. Крім того треба усвідомлювати що ряд обмежень цілісності закладає сам програміст і саме на цьому кроці вже є відповідальність за забезпечення захисту даних, бо неправильні обмеження, які не відповідають реальним потребам предметної області, вже порушать цілісність, навіть якщо їх дотримувати. Найпростіший приклад, якщо поле потребує довжини 300 символів, а дозволено лиш 250, деякі дані будуть втрачатися.

Згідно з визначенням у [5], цілісність реляційних даних забезпечується зовнішніми ключами в дочірніх відношеннях. Дочірніми називаються відношення, де присутні зовнішні ключі, значення яких повинні посилатися на значення потенційних ключів у батьківських відношеннях.

Посилальна цілісність забезпечується, якщо значення зовнішнього ключа дочірнього відношення посилається на існуюче значення потенційного ключа батьківського відношення, або задається за допомогою визначника NULL.

Допустимість наявності визначника NULL серед значень атрибутів, що використовуються як зовнішній ключ, встановлюється вимогами користувачів. [5] — від правильності прийнятого рішення залежить надійність подальшої робота з базою. Вже на цьому етапі маємо не лише правильно закладати тип поля, щоб такий зовнішній ключ створився, але і враховувати можливість побудови зв'язку між таблицями таким чином, щоб СКБД забезпечувала захист від некоректної роботи та порушення цілісності даних. Зрозуміло, якщо ми, при побудові зовнішніх зв'язків між таблицями, закладемо каскадне вилучення взаємопов'язаних записів — тим самим створимо прогалину у захисті інформації на рівні СКБД.

ISSN 2663 - 4023

Нижче (рис. 2) продемонстровано, як застосовуються обмеження цілісності атрибутів при проектування зовнішніх зв'язків між таблицями — батьківське поле має бути або ключовим (PRIMARY KEY) або унікальним (UNIQUE) і батьківське та дочірнє мають співпадати за типом та розміром. Наприклад, у таблицях type_opetration та operations, для яких встановлено зовнішній звязок id (type_opetration — батьківська таблиця) та id_typeoperation (operations — дочірня таблиця).

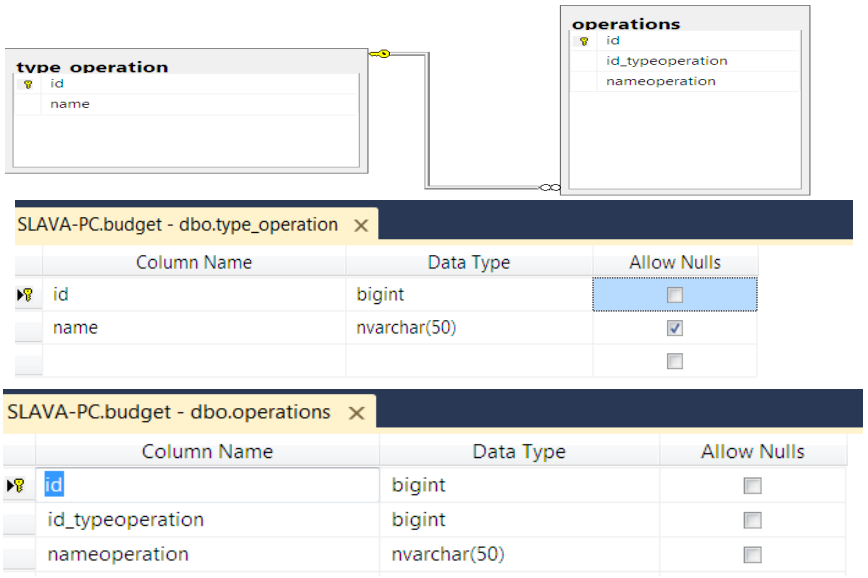

Рис. 2. Обмеження цілісності атрибутів при проектуванні зовнішніх зв'язків між таблицями

Задаючи зв'язки через зовнішні ключі, ми маємо змогу використати СКБД для захисту від некоректного вилучення чи оновлення запису через неуважність чи через злий умисел. СКБД за замовчуванням забороняє такі дії, якщо це стосується батьківського запису, на який є посилання у кортежах дочірніх таблиць. Проте у конструкції ключа можна задати іншу поведінку щодо записів у дочірніх таблицях при вилученні та/або оновленні запису у батьківській таблиці.

Проілюструємо це на прикладі таблиць MS SQL Server. Між таблицями задано зовнішній зв'язок (рис. 2) і дочірня таблиця має дані, як зображено на рис. 3.

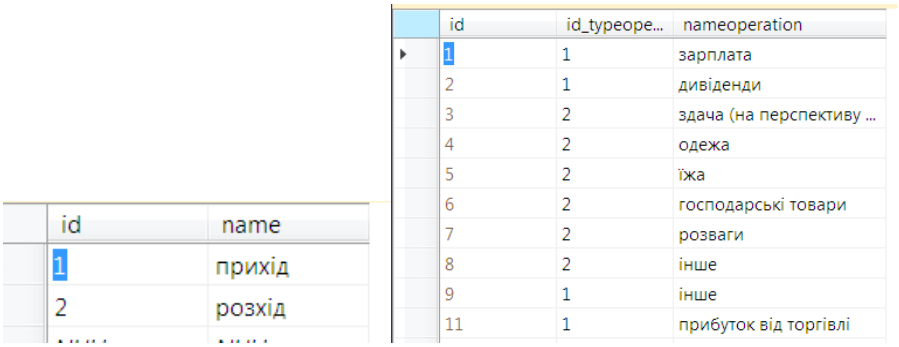

Рис. 3. Приклад заповнення ключових полів та даних для батьківської та дочірньої таблиць пов'язаних зовнішнім зв'язком

При таких обмеженнях СКБД не дозволить вилучити батьківський запис.

ISSN 2663 - 4023

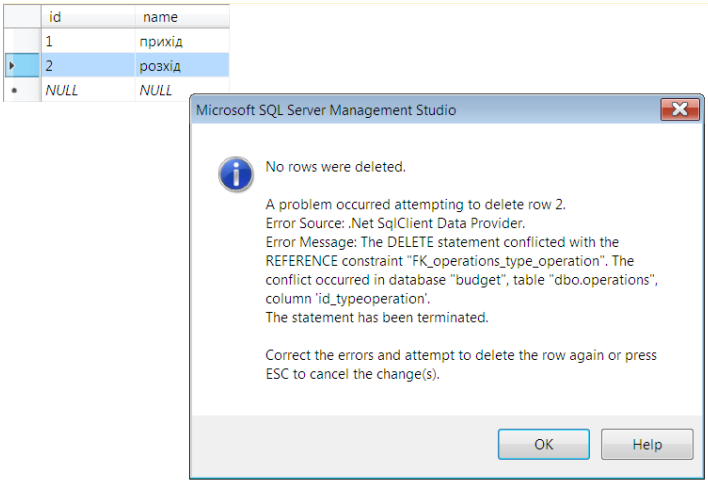

Рис. 4. Повідомлення СКБД при спробі некоректного вилучення батьківського запису при наявності даних у дочірній таблиці

На етапі реалізації зв'язків можна також протидіяти некоректному вводу даних у дочірню таблицю, роблячи обов'язковим поле посилання на запис у батьківській таблиці.

Оскільки у таблиці operations поле id_typeoperation задано NOT NULL (див. рис. 2) то при спробі обавити запис який не має батьківського і порушує цілісність даних СКБД попередить і не дасть виконати таку дію як показано на рис. 4.

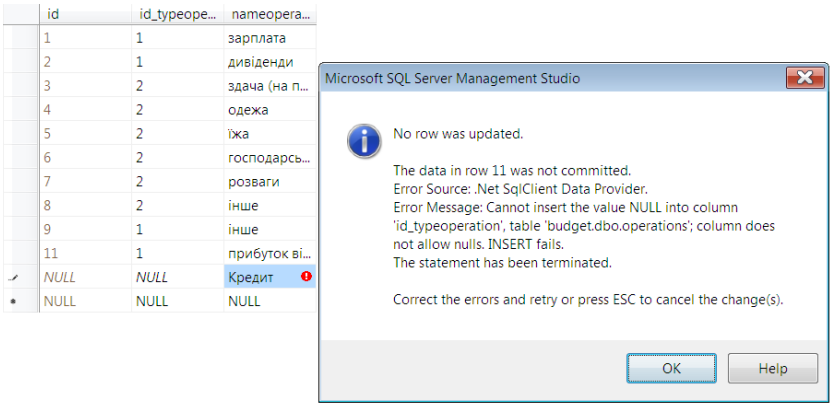

Рис. 5. Повідомлення СКБД при спробі некоректної внесення дочірнього запису з порушенням обмеження NOT NULL для поля яке слугує для зовнішнього звязку з батьківською таблицею

3. Нормалізація та реляційні зв'язки як метод захисту даних. Нормалізуючи базу даних (2 та 3 нормальна форма) та задаючи зв'язки між таблицями реляційної бази даних, ми в таблиці вводимо поля-посилання на ключі батьківської таблиці, замість самих даних. Цим самим утруднюється крадіжка інформації з бази даних, бо в таблиці є коди полів, а вміст знаходиться у відповідних таблицях-довідниках.

Крім того, при потребі, створюючи таблиці та поля в базі даних, закодовують назви так, щоб вони стали анонімними і важко було орієнтуватися про вміст таблиці чи поля і лиш ПЗ, яке працює з базою даних, показує дані користувачу і відображає назву, зручну для інтерфейсу користувача.

ISSN 2663 - 4023

На рис. 6 наведено приклад: таблицю реляційної бази даних з зашифрованими назвами полів, яка містить посилання на інші довідники та таблиці. Зрозуміло, що тут побудова зв'язків між декількома таблицями допомагає приховати дані від несанкціонованого перегляду та викрадення.

	REC	F1_	F1_MI	F9_M	F9_MI
1	1802DBF3EA9BB	474CD925B8CE	F8317C73F79B	000000000000	000000000000
2	38925F32099EE	6DCFAB3240D7	C8D749208C9B	50B77DBD5FD	807A15B25052
3	D8AB6A1BA89BE	474CD925B8CE	184DE42BCA9B	50B77DBD5FD	807A15B25052
4	F0730B04A99FB	474CD925B8CE	086E0A6F629C	000000000000	000000000000

Рис. 6. Таблиця реляційної бази даних з зашифрованими назвами полів

4. Внутрішні загрози при вибірці даних та методи усунення. Доцільно створювати view та дозволяти клієнтським програмам працювати з ними, а не безпосередньо з таблицями — це дасть подвійний захист:

- на етапі надання дозволу на доступ — клієнт не зможе доступитися до таблиці, а лише до її view;
- від клієнта можна приховати поля, які є ключовими або використовуються для захисту даних — ці поля мають бути приховані від стороннього доступу та перегляду.

При вибірці даних треба уникати дуже громіздких запитів та view, адже можуть виникнути зависання та збої через вихід часу на обробку. Через таку проблему вибірки даних виникає вразливість, яка аналогічна описаній у [9] (подібно до DDoS атак, які приводять до відмови в обслуговуванні, коли кіберзлочинець «завалює» цільовий сервер, після чого той не має можливості виконувати легальні запити від реальних користувачів). В основному, сервер стає недоступним або дає збій — у випадку проблем з вибірками даних це відбувається через внутрішню причину. Тому варто оптимізувати запити, використовувати тимчасові таблиці або серверні процедури. При звертанні до вибірки з програмного коду, щоб запобігти аварійному перериванню запиту до бази, можна також програмно збільшувати час очікування (timeout), але краще, щоб вибірка цього не потребувала.

Серед кращих практик розробки, ефективних з точки зору оптимізації запитів, є використання з'єднання вибірок (підвибірок) (операція мови SQL — JOIN) замість підзапитів. Так, навіть критерії або матриці компетентності для відбору кандидатів на роботу компаній розробників ПЗ зазвичай передбачають наявність таких знань та навичок.

Розглянемо приклади.

1. Треба визначити номер, виробника і ціну найдорожчого виробу, іншими словами, знайти рядок, що містить максимальне значення деякого стовпця. Це не варто робити за допомогою вкладеного запиту:

```
SELECT kod_article, produser, price
FROM order
WHERE price=(SELECT MAX(price) FROM order)
```

Той самий результат, але з кращим планом виконання можна отримати, використавши join:

```
SELECT Max_art.kod_article, dbo.shop. produser, dbo.shop.price
FROM (SELECT MAX(kod_article) AS article
FROM dbo.order) Max_art INNER JOIN
dbo.order ON Max_art.kod_article = dbo.order.kod_article
```

ISSN 2663 - 4023

2. Як оптимізувати підрахунок кількості студентів для кожного наказу на зарахування у списку наказів order, якщо перелік студентів міститься у дочірній таблиці newstudent_item?

```
SELECT *, (SELECT COUNT(order_id) FROM dbo.newstudent_item AS
item WHERE (dbo.[order].id = order_id)) AS countitem FROM dbo.[order]
```

Внесемо невелику кількість даних для наочності:

order

id	order_type	speciality_id	actual
1	1	121	1
2	1	121	1
3	1	125	1
4	1	125	1

newstudent_item

id	order_id	student_id	student_ident	enronment_id
1	1	1245	244	255
2	1	1247	246	275
3	1	1047	46	605
4	3	1147	146	495
5	4	1100	99	173

Запит з кращим планом виконання, який дає виграш у часі та ресурсах на великій кількості даних:

```
SELECT dbo.[order].*, ISNULL(item.countitem, 0) AS countitem
FROM dbo.[order] LEFT OUTER JOIN
    (SELECT order_id AS id, COUNT(id) AS countitem
     FROM dbo.newstudent_item
     GROUP BY order_id) AS item ON dbo.[order].id = item.id
```

Результати обидвох запитів міститимуть дані про всі накази та відповідну кількість студентів, навіть якщо в дочірній таблиці нема студентів:

id	order_type	speciality_id	actual	countitem
1	1	121	1	3
2	1	121	1	0
3	1	125	1	1
4	1	125	1	1

Або запит для відображення лише наказів, для яких існують дані про студентів у дочірній таблиці:

```
SELECT dbo.[order].*, item.countitem
FROM dbo.[order] INNER JOIN
(SELECT order_id AS id, COUNT(id) AS countitem
FROM dbo.ordernewstudentitem
GROUP BY order_id) AS item ON dbo.[order].id = item.id
```

id	order_type	speciality_id	actual	countitem
1	1	121	1	3
3	1	125	1	1
4	1	125	1	1

5. Специфікація прав доступу та дотримання цілісності. При програмуванні ведення баз даних та будь-яких робіт з базами даних через прикладну програму крім питання встановлення безпечного зв'язку з базою даних, яке на даному етапі не розглядаємо, слід пам'ятати про необхідність специфікації права доступу. Вже на рівні

СКБД має бути налаштовано захист через права користувача та/або належність користувача до ролі для [1]:

- створення бази даних;
- означення та переозначення видалення таблиці або віртуальної таблиці;
- вибірки додавання видалення оновлення рядків у таблиці або віртуальній таблиці;
- виконання збережених процедур.

Від того, наскільки правильно змодельовано область прикладної задачі та від закладання умов цілісності залежить правильність подальшої роботи з даними і збереження їх достовірності при проведенні операцій. Дієвим засобом, котрий контролює коректність змін і забезпечує дотримання цілісності, в тому числі семантичної, на боці СКБД є тригери — вони реагують на проведення операцій та захищають від некоректних змін або проводять зміни згідно умов дотримання цілісності, закладеної для конкретної бази даних та допомагають вести додатковий облік змін для адміністрування баз даних.

За дотримання логічного ланцюжка змін у базі даних можуть відповідати серверні функції та процедури. Вони не доєднані до таблиці чи представлення (view), як тригери, а викликаються програмним застосунком чи іншою серверною процедурою.

6. Проектування і розробка ПЗ для баз даних та засоби безпечного коду для роботи з даними. Програмне забезпечення має теж розроблятися з урахуванням прав на операції, для яких специфікуються права доступу на рівні СКБД. На боці ПЗ за дотриманням обмежень цілісності атрибутів слідкують також валідатори — це можуть бути і процедури, які працюють і на боці клієнта, і на боці сервера так і готові контроли (наприклад в bootstrap). Валідатори, особливо серверні, не лише контролюють дані, що вносяться в базу, на відповідність обмеженням цілісності, але також можуть робити перевірку на наявність шкідливих команд у надісланих клієнтською програмою параметрах запиту. Про доцільність параметризованих запитів та валідації для запобігання т. зв. SQL ін’єкціям говориться також у [12]. Зазначимо, що в цьому розумінні ефективно застосовувати уже підготовлені шаблони — заповнювачі (Placeholders) SQL запиту. Їх можна розуміти як попередньо визначені шаблони, що будуть замінені деякими фактичними значеннями під час виконання. Сервер спочатку обробляє шаблон SQL виразу, опісля отримує фактичні значення, валідує їх і використовує безпосередньо в момент запуску запиту на виконання (вже після того, як був сформований сам запит). Таким чином загроза виконання сервером шкідливого вкраплення усувається.

Ще одним методом захисту цілісності даних є використання транзакцій. Транзакції враховують моменти, коли одна логічна операція для збереження цілісності даних полягає у виконанні кількох кроків — запитів до БД, якщо один з них не виконано коректно, слід відмінити всі решта, які з ними пов’язані. Наприклад, доцільно поміщати в одну транзакцію занесення даних про виконання замовлення і відвантаження товару. Зрозуміло, що кількість товару на складі при відвантаженні зменшиться, і має бути внесено одночасно зміни і у таблицю, яка відображає, куди він пішов, і в таблицю, яка показує наявність (скільки ще залишилося товару), інакше виникне невідповідність у балансі. Вимоги до проведення транзакцій ACID (атомарність, узгодженість, ізольованість, надійність) широко висвітлені в професійній літературі, тому в даному матеріалі не зупиняємося на них детально, лиш зазначаємо, що дотримання принципів ACID є обов’язковим кроком для забезпечення захисту даних при роботі з базами даних.

Надійність розробленого ПЗ полягає в тому, щоб на рівні прикладної програми контролювати та випереджувати ситуації, коли СКБД дає повідомлення про неможливість виконання операції через порушення цілісності або відсутність прав у користувача — це має робити інтерфейс прикладної програми.

ПЗ повинне проводити перевірку і не надсилати запит на виконання такої дії до БД. Найпростіший приклад реалізації такої перевірки — коли користувача відкритого інтернет ресурсу відсилають на сторінку логіну, чи пропонують отримати логін та пароль перед тим, як оформити замовлення. Тут допуск до відповідних ресурсів та дій з ними мають лише авторизовані користувачі. Найчастіше дія полягає у надсиланні запиту на сервер на додавання записів у таблицю замовлень згідно з заповненою користувачем формою. Якщо менеджер вносить якісь дані про товари, програмне забезпечення не дозволить йому внести запис про новий товар, якщо він не обере групу товарів, до якої товар належить. У випадку відсутності такої групи, пропонується її створити — таким чином на рівні реалізації ПЗ проводиться контроль цілісності зовнішніх зв’язків між батьківською таблицею (груп товарів) та дочірньою (товарів). По суті це виконання обмеження not null на поле зовнішнього ключа, але СКБД не отримає некоректного запиту, бо ПЗ реалізує таку перевірку. Відповідь на рівні СКБД проілюстрована на рис 5. При спробі вилучити групу товарів, щодо якої існують записи у довіднику товарів чи у замовленнях, програма попередить, що така дія неможлива — це реалізація на рівні програмного забезпечення захисту цілісності зовнішніх зв’язків між батьківською таблицею груп товарів та дочірніми товарів та замовлень. Якби такий запит надійшов до СКБД, спрацював би захист constraint on delete none по замовчуванню, як на рис 5. Але ПЗ має перехопити цю ситуацію і попередити користувача.

Якщо розробляється програмне забезпечення для адміністрування, з міркувань безпеки варто приховувати шлях до програми «адмінки» від звичайного користувача, навіть якщо він має пароль і права доступу до бази даних. Тут доцільно передбачити ще захист паролем доступу до самої програми адміністрування і, якщо середовище розробки це дозволяє, наприклад, не висвічувати у браузері шлях до програми та її назву. Стаття не має на меті опис загроз для web застосунків, тому щодо цього питання лиш зазначимо, що і ПЗ для адміністрування і ПЗ для роботи з клієнтами потребують грамотної маршрутизації та запобігання перенаправленню на шкідливу сторінку.

В залежності від призначення бази даних приймається рішення, чи надавати неавторизованому користувачу доступ до перегляду інформації бази даних. Якщо іде мова про інтернет-магазин, то доцільно такий доступ надавати; у випадку внутрішніх баз даних підприємств та організацій допуск, навіть для перегляду, доцільно надавати лише авторизованими користувачам або захищати паролем доступ до програмного забезпечення з такими можливостями.

Щодо програмного забезпечення для працівників. котрі мають права доступу і виконують роботи з деякими таблицями БД, доцільно розробити для них ролі наприклад «менеджер», «бухгалтер», «керівник», в яких специфікувати перелік таблиць, до яких вони мають доступ, та права на виконання дій з ними і відносити користувачів до певної ролі, щоб не прописувати для кожного окремо права доступу у БД.

Вже для авторизованих користувачів, відповідно до можливостей середовища розробки та самої задачі, програміст приймає рішення про реалізацію стратегії допусків до виконання тих чи інших дій, щодо яких специфікуються права доступу.

Наприклад, прикладна програма може робити перевірку, чи авторизований користувач і до якої ролі він належить і після цього скеровувати його на сторінку, відповідно до його ролі (прав). Або при генерації сторінки враховувати роль —

наприклад: для кожної ролі створювати своє інтерфейсне меню з допомогою заготовки-шаблону або не показувати якісь пункти меню чи кнопки, що відповідають за доступ до таблиці чи дію, на яку поточний користувач не має прав.

Окрім того (при потребі такого захисту) дія видалення звичайним користувачем запису може прикладною програмою трактуватися не як фізичне знищення запису, а як переведення у інший стан — наприклад, запис стає неактивним і невидимим для програм перегляду і лише адміністратор приймає рішення чи вилучати записи фізично.

При програмуванні операції вилучення запису зазвичай роблять захист від неуважності користувача — випадкового натискання кнопки чи клавіші, яка відповідає за дію вилучення запису. У такому випадку, як правило, виводиться повідомлення з проханням підтвердити намір виконання такої дії.

Важливими аспектами, що відповідають меті створення безпечного коду при взаємодії ПЗ з базами даних, є шифрування та запобігання SQL ін’єкціям. Детально не зупиняємося на цій темі, лиш висвітлюємо деякі аспекти, на які слід звернути увагу.

Щодо шифрування зазначимо, що варто застосовувати і шифрування самих даних (тут слід окремо відзначити TDE сертифікати які дають захист від копіювання бази), і шифрування трафіку даних — застосування TLS/SSL сертифікатів для передачі даних. Перехоплення SQL ін’єкцій важливе як для запобігання відкритим атакам, так і для боротьби з «тихою» крадіжкою даних.

Деякі нові технології та платформи для розробки мають певні вбудовані елементи такого захисту (наприклад, LINQ to SQL, HTTPS).

У розділах статті, присвячених розробці ПЗ, не наводяться приклади коду, оскільки описані підходи стосуються всіх програмних застосунків для роботи з базами даних, незалежно від мови розробки та середовища реалізації. Але слід наголосити: якщо використовуємо легко вразливу web технологію на зразок php+mysql, слід приділяти особливу увагу аспектам захисту, зокрема від атак через SQL-ін’єкції.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Якщо на етапі проектування бази даних при програмуванні структури та зв’язків бази даних ми закладаємо підвалини безпеки даних, то на етапі проектування та конструювання ПЗ професійний підхід слугує забезпеченню цілісності, достовірності та захисту від некоректних запитів до бази даних на кожному етапі роботи ПЗ з даними.

Від професійності проектування реляційних баз даних, зокрема нормалізації, структури таблиць та зв’язків, закладених у проекті обмежень цілісності та засобів для виконання цих обмежень, в т.ч. тригерів, залежить ефективність захисту даних з боку СКБД. Лише після цього можна говорити про дотримання цілісності та достовірності на подальших етапах розробки та експлуатації ПЗ, в т.ч. про розробку безпечного коду та залучення спеціальних засобів захисту даних на подальших кроках життєвого циклу розробки ПЗ та його експлуатації.

Тому варто розширювати поняття захисту даних на область проектування реляційних баз даних, а підхід «безпечний код» повинен включати також етапи проектування та розробки самих баз даних. Деякі розробники не усвідомлюють або нехтують цим, що є виявом непрофесіоналізму. Тому в заголовку статті наголошено, що професійний підхід дає захист при розробці реляційних баз даних. Підхід «безпечний код» є серед заходів інформаційної безпеки при виготовленні сертифікованих програмних продуктів, отже доцільно та ефективно застосовувати його не лише на усіх

етапах життєвого циклу розробки ПЗ для роботи з реляційними базами даних, але і на етапах проектування та розробки самих баз даних.

Прогнозується систематизувати викладений матеріал для формування критеріїв, що застосовуються у проектуванні, конструюванні баз даних та ПЗ і при оцінюванні якості розробок та їх загрозостійкості.

ПОДЯКА

Д.т.н., професору, завідувачу кафедри захисту інформації Івану Опірському за поради у науковому пошуку та оформленні статті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Безпека баз даних – Datalabs.* (б.д.). Datalabs. https://datalabsua.com/ua/database-security/.
2. Pernul, G., Tjoa, A., & Winiwarter, W. (1998). Modelling Data Secrecy and Integrity. *Data & Knowledge Engineering, 26(3),* 291–308. https://doi.org/10.1016/s0169-023x(97)00045-1
3. Yesin Y., et al. (2021). Ensuring Data Integrity in Databases with the Universal Basis of Relations. *Appl. Sci. 11, 8781*. https://doi.org/10.3390/app11188781
4. Martin, R. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Pearson.
5. *Поняття логічної і фізичної цілісності даних.* (б.д.). https://jak.bono.odessa.ua/articles/ponjattja-logichnogo-i-fizichnoi-cilisnosti-danih.php
6. *Secure Coding - Challenges in information security*. (n.d.). DQS|Audits und Zertifizierung|Simply leveraging Quality. https://www.dqsglobal.com/intl/learn/blog/secure-coding-challenge-in-information-security
7. *Організація баз даних та знань. Тема 8 – Цілісність та безпека даних. Конспект лекції.* (б.д.). Elearning SumDU. https://elearning.sumdu.edu.ua/free_content/lectured:89b3d175c06a6b137e410cb14821d0e94549ad5a/20151013153156/44700/index.html#p22
8. Vakhula, O., & Opirskyy, I. (2023). Research on Security Issues in Cloud Environments and Solutions Using the "Security as Code" Approach. *Ukrainian Inf. Secur. Res. J. 25(3),* 113–122. https://doi.org/10.18372/2410-7840.25.17936
9. Касянчук, Н., & Ткачук, Л. (2019). Захист інформації в базах даних. *Вінницький національний технічний університет.* https://conferences.vntu.edu.ua/index.php/all-fm/all-fm-2019/paper/download/7001/5715
10. Faragallah, O., et al. (2014). Multilevel Security for Relational Databases. *Auerbach Publications*. https://doi.org/10.1201/b17719
11. *Secure software development methodologies|BETWEEN Technology*. (n.d.). Impúlsate|blog de BETWEEN Technology. https://impulsate.between.tech/en/secure-dsoftware-development-methodologies
12. Teimoor, R. (2021). A review of database security concepts, risks, and problems. *UHD Journal of Science and Technology, 5(2),* 38–46. https://doi.org/10.21928/uhdjst.v5n2y2021.pp38-46

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (23), 2024
ISSN 2663 - 4023

Yaroslava Momryk
Assistant of Department of Information Security
Lviv Politecnic National University, Lviv, Ukraine
ORCID 0009-0001-4114-0347
sm.slyala@gmail.com

Yuriy Yashchuk
Ph. D., Assoc. Prof., Head of Department of Applied Mathematics
Ivan Franko National University of Lviv, Lviv, Ukraine
ORCID 0000-0003-4935-497X
yuriy.yashchuk@lnu.edu.ua

Roman Tuchapskyi
Candidate of physical and mathematical sciences, senior researcher
Pidstrygach Institute of Applied Problems in Mechanics and Mathematics, National Academy of Sciences of Ukraine, Lviv, Ukraine
ORCID 0000-0002-2556-1088
roman.tuch@gmail.com

A PROFESSIONAL APPROACH AS A METHOD OF PROTECTING INFORMATION AT THE STAGES OF DEVELOPMENT OF RELATIONAL DATABASES AND SOFTWARE FOR WORKING WITH THEM

Abstract. The design of relational databases and software (software) for working with them is analyzed from the point of view of design components affecting data security. Named the internal threats that arise due to imperfect design. Positive practices are described that allow designing databases and developing software to work with them in the aspect of secure code. It is justified why the stage of relational database design, when relation normalization is applied, relationships between tables are formed, and integrity constraints are formed, is a step to ensure data protection, which is not emphasized in the database security literature. In particular, it is proposed to create external relationships between tables so that the Database Management System (DBMS) protects data from integrity violations. Some points in the development of software are highlighted, which are responsible for the safety of working with the database from the point of view of creating a reliable and safe code and stem from the practical experience of the programmer. The secure code approach, which is widely used in software development and at the software audit level, has been shown to prevent internal security threats, which are the most common cause of data loss. The requirements for using this approach are included in the updated information security standards, so they should be considered by professional database and software developers.

Keywords: code security; data security; database design; software engineering; security threat; data integrity; integrity constraints; database normalization.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. *Database Security – Datalabs.* (n.d.). Datalabs. https://datalabsua.com/ua/database-security/
2. Pernul, G., Tjoa, A., & Winiwarter, W. (1998). Modelling Data Secrecy and Integrity. *Data & Knowledge Engineering, 26(3),* 291–308. https://doi.org/10.1016/s0169-023x(97)00045-1
3. Yesin Y., et al. (2021). Ensuring Data Integrity in Databases with the Universal Basis of Relations. *Appl. Sci. 11, 8781*. https://doi.org/10.3390/app11188781
4. Martin, R. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Pearson.
5. *Concept of logical and physical data integrity.* (n.d.). https://jak.bono.odessa.ua/articles/ponjattja-logichnogo-i-fizichnoi-cilisnosti-danih.php
6. *Secure Coding - Challenges in information security*. (n.d.). DQS|Audits und Zertifizierung|Simply leveraging Quality. https://www.dqsglobal.com/intl/learn/blog/secure-coding-challenge-in-information-security

7. *Databases and knowledge bases design. Chapter 8 – Data integrity and security*. *Lecture notes.* (n.d.). Elearning SumDU. https://elearning.sumdu.edu.ua/free_content/lectured:89b3d175c06a6b137e410cb14821d0e94549ad5a/20151013153156/44700/index.html#p22
8. Vakhula, O., & Opirskyy, I. (2023). Research on Security Issues in Cloud Environments and Solutions Using the "Security as Code" Approach. *Ukrainian Inf. Secur. Res. J. 25(3),* 113–122. https://doi.org/10.18372/2410-7840.25.17936
9. Kasianchuk, N., & Tkachuk, L. (2019). Information protection in databases. *Vinnytsia National Technical University*. https://conferences. vntu. edu. ua/index. php/all-fm/all-fm-2019/paper/download/7001/5715
10. Faragallah, O., et al. (2014). Multilevel Security for Relational Databases. *Auerbach Publications*. https://doi.org/10.1201/b17719
11. *Secure software development methodologies|BETWEEN Technology*. (n.d.). Impúlsate|blog de BETWEEN Technology. https://impulsate.between.tech/en/secure-dsoftware-development-methodologies
12. Teimoor, R. (2021). A review of database security concepts, risks, and problems. *UHD Journal of Science and Technology, 5(2),* 38–46. https://doi.org/10.21928/uhdjst.v5n2y2021.pp38-46